(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2000－59788
（P2000－59788A）
(43)公開日　平成12年２月25日(2000.2.25)

(51)Int.Cl.[7]	識別記号	FI		テーマコード(参考)
H04N 7/32		H04N 7/137	Z	5C059
7/24		7/13	Z	
7/30		7/133	Z	

審査請求　未請求　請求項の数３　ＯＬ　（全 62 頁）

(21)出願番号　特願平10－200160

(22)出願日　平成10年７月15日(1998.7.15)

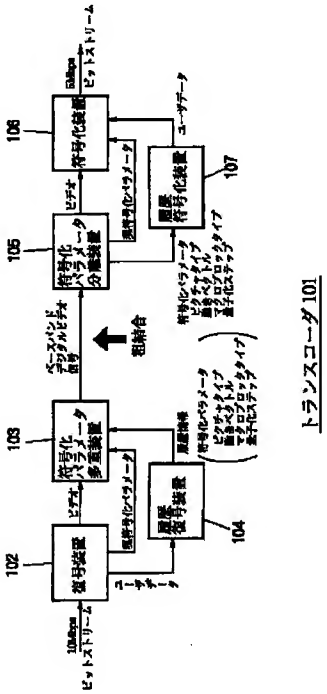
(31)優先権主張番号　特願平10－58118
(32)優先日　平成10年３月10日(1998.3.10)
(33)優先権主張国　日本（ＪＰ）
(31)優先権主張番号　特願平10－157245
(32)優先日　平成10年６月５日(1998.6.5)
(33)優先権主張国　日本（ＪＰ）

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号
(72)発明者　田原　勝己
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(72)発明者　北村　卓也
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(74)代理人　100082131
弁理士　稲本　義雄

最終頁に続く

(54)【発明の名称】　復号装置および方法、並びに提供媒体

(57)【要約】
【課題】　トランスコーダの規模を小さくするとともに、再符号化に伴う画像の劣化を抑制する。
【解決手段】　符号化パラメータ多重装置１０３は、現符号化パラメータと履歴復号装置１０４より供給される履歴情報に含まれる複数世代の符号化パラメータとを、復号装置１０２より供給されるビデオデータに多重化し、ベースバンドのデジタルビデオ信号として符号化パラメータ分離装置１０５に出力する。符号化パラメータ分離装置１０５は、符号化装置１０６で符号化に使用する符号化パラメータを選択し、現符号化パラメータとして符号化装置１０６に出力するとともに、残りの複数世代の符号化パラメータを履歴符号化装置１０７に出力する。符号化装置１０６は、符号化パラメータ分離装置１０５より供給されるビデオデータを現符号化パラメータで符号化してビットストリームを生成すると共に、そのビットストリームに履歴符号化装置１０７より供給される複数世代の符号化パラメータが履歴情報として含まれているユーザデータを多重化し、後段のトランスコーダに出力する。



【特許請求の範囲】
【請求項１】　MPEG規格に基づいて符号化されているビットストリームを復号する復号装置において、
前記ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号手段と、
前記ビットストリームからビデオデータを復号するビデオデータ復号手段とを備えることを特徴とする復号装置。
【請求項２】　MPEG規格に基づいて符号化されているビットストリームを復号する復号装置の復号方法において、
前記ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号ステップと、
前記ビットストリームからビデオデータを復号するビデオデータ復号ステップとを含むことを特徴とする復号方法。
【請求項３】　MPEG規格に基づいて符号化されているビットストリームを復号する復号装置に、
前記ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号ステップと、
前記ビットストリームからビデオデータを復号するビデオデータ復号ステップとを含む処理を実行させるコンピュータが読み取り可能なプログラムを提供することを特徴とする提供媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、復号装置および方法、並びに提供媒体に関し、特に、動画像信号を、例えば光磁気ディスクや磁気テープなどの記録媒体に記録し、これを再生して、ステレオ視が可能なディスプレイなどに表示したり、テレビ会議システム、テレビ電話システム、放送用機器など、動画像信号を伝送路を介して送信側から受信側に伝送し、受信側において、これを受信して表示する場合などに用いて好適な復号装置および方法、並びに提供媒体に関する。
【０００２】
【従来の技術】例えば、テレビ会議システム、テレビ電話システムなどのように、動画像信号を遠隔地に伝送するシステムにおいては、伝送路を効率良く利用するため、映像信号のライン相関やフレーム間相関が利用されて、画像信号が圧縮符号化される。
【０００３】画像信号が圧縮符号化される場合、生成されるビットストリームが、所定のビットレートになるように符号化が行われる。しかしながら、実運用上において、伝送路の都合により、ビットストリームのビットレートを変換する必要が生じることがある。このような場合、図６８に示すようなトランスコーダ１３１により、符号化されている情報を一旦復号し、ビットレートが所定の値になるように、再び符号化する方法が一般的である。図６８の例の場合、１０Mbpsで送られてきたビットストリームが、復号装置１３２により復号され、デジタルビデオ信号として符号化装置１３３に供給され、符号化装置１３３により、ビットレートが５Mbpsであるビットストリームに符号化されて出力される。
【０００４】
【発明が解決しようとする課題】このように映像信号を再符号化する場合、符号化装置１３３には、図６９に示すように、映像信号のライン相関やフレーム間相関を検出する動き検出部１３４が必要となり、符号化装置１３３の規模が大きくなる課題があった。
【０００５】また、例えば放送局においては、映像の編集が秒単位で行われるので、フレームの画像情報が他のフレームの画像情報と独立しているほうがよい。そこで、図７０に示すように、低いビットレート（３乃至９Mbps）で転送しても画質が劣化しないように、情報が相関関係にあるフレームの集合であるGOP(Group of Picture)を構成するフレーム数が多いLong GOPの符号化装置１３３－１から出力されたビットストリームは、放送局の符号化装置１３３－２により、GOPを構成するフレーム数が少ないShort GOPに変換されて高ビットレート（１８乃至５０Mbps）で伝送され、編集終了後、符号化装置１３３－３により、再度Long GOPに変換されて出力される。このように、画像情報に符号化、復号が繰り返されると、符号化の度に使用される符号化パラメータが変化するので画像情報が劣化する課題があった。
【０００６】本発明はこのような状況に鑑みてなされたものであり、過去に演算した動きベクトルを用いて再符号化を行うことにより、装置の規模を小さくするとともに、再符号化に伴う画像の劣化を抑制することを可能とするものである。
【０００７】
【課題を解決するための手段】請求項１に記載の復号装置は、ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号手段と、ビットストリームからビデオデータを復号するビデオデータ復号手段とを備えることを特徴とする。
【０００８】請求項２に記載の復号方法は、ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号ステップと、ビットストリームからビデオデータを復号するビデオデータ復号ステップとを含むことを特徴とする。
【０００９】請求項３に記載の提供媒体は、ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号ステップと、ビットストリームからビデ

オデータを復号するビデオデータ復号ステップとを含む処理を実行させるコンピュータが読み取り可能なプログラムを提供することを特徴とする。

【００１０】請求項１に記載の復号装置、請求項２に記載の復号方法、および請求項３に記載の提供媒体においては、ビットストリームのピクチャ層のユーザデータエリアに挿入された、符号化履歴情報が復号される。

【００１１】

【発明の実施の形態】以下に本発明の実施の形態を説明するが、特許請求の範囲に記載の発明の各手段と以下の実施の形態との対応関係を明らかにするために、各手段の後の括弧内に、対応する実施の形態（但し一例）を付加して本発明の特徴を記述すると、次のようになる。

【００１２】請求項１に記載の復号装置は、ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号手段（例えば、図１５の履歴復号装置１０４）と、ビットストリームからビデオデータを復号するビデオデータ復号手段（例えば、図１５の復号装置１０２）とを備えることを特徴とする。

【００１３】但し勿論この記載は、各手段を記載したものに限定することを意味するものではない。

【００１４】本発明を適用したトランスコーダについて説明する前に、動画像信号の圧縮符号化について説明する。なお、本明細書においてシステムの用語は、複数の装置、手段などにより構成される全体的な装置を意味するものである。

【００１５】例えば、テレビ会議システム、テレビ電話システムなどのように、動画像信号を遠隔地に伝送するシステムにおいては、伝送路を効率良く利用するため、映像信号のライン相関やフレーム間相関を利用して、画像信号を圧縮符号化するようになされている。

【００１６】ライン相関を利用すると、画像信号を、例えばDCT（離散コサイン変換）処理するなどして圧縮することができる。

【００１７】また、フレーム間相関を利用すると、画像信号をさらに圧縮して符号化することが可能となる。例えば図１に示すように、時刻ｔ１乃至ｔ３において、フレーム画像ＰＣ１乃至ＰＣ３がそれぞれ発生している場合、フレーム画像ＰＣ１およびＰＣ２の画像信号の差を演算して、ＰＣ１２を生成し、また、フレーム画像ＰＣ２およびＰＣ３の差を演算して、ＰＣ２３を生成する。通常、時間的に隣接するフレームの画像は、それ程大きな変化を有していないため、両者の差を演算すると、その差分信号は小さな値のものとなる。そこで、この差分信号を符号化すれば、符号量を圧縮することができる。

【００１８】しかしながら、差分信号のみを伝送したのでは、元の画像を復元することができない。そこで、各フレームの画像を、Ｉピクチャ、ＰピクチャまたはＢピクチャの３種類のピクチャタイプのいずれかとし、画像信号を圧縮符号化するようにしている。

【００１９】すなわち、例えば図２に示すように、フレームＦ１乃至Ｆ１７までの１７フレームの画像信号をグループオブピクチャ(GOP)とし、処理の１単位とする。そして、その先頭のフレームＦ１の画像信号はＩピクチャとして符号化し、第２番目のフレームＦ２はＢピクチャとして、また第３番目のフレームＦ３はＰピクチャとして、それぞれ処理する。以下、第４番目以降のフレームＦ４乃至Ｆ１７は、ＢピクチャまたはＰピクチャとして交互に処理する。

【００２０】Ｉピクチャの画像信号としては、その１フレーム分の画像信号をそのまま伝送する。これに対して、Ｐピクチャの画像信号としては、基本的には、図２に示すように、それより時間的に先行するＩピクチャまたはＰピクチャの画像信号からの差分を伝送する。さらにＢピクチャの画像信号としては、基本的には、図３に示すように、時間的に先行するフレームまたは後行するフレームの両方の平均値からの差分を求め、その差分を符号化する。

【００２１】図４は、このようにして、動画像信号を符号化する方法の原理を示している。同図に示すように、最初のフレームＦ１は、Ｉピクチャとして処理されるため、そのまま伝送データＦ１Ｘとして伝送路に伝送される（画像内符号化）。これに対して、第２のフレームＦ２は、Ｂピクチャとして処理されるため、時間的に先行するフレームＦ１と、時間的に後行するフレームＦ３の平均値との差分が演算され、その差分が伝送データＦ２Ｘとして伝送される。

【００２２】ただし、このＢピクチャとしての処理は、さらに細かく説明すると、４種類存在する。その第１の処理は、元のフレームＦ２のデータをそのまま伝送データＦ２Ｘとして伝送するものであり（ＳＰ１）（イントラ符号化）、Ｉピクチャにおける場合と同様の処理となる。第２の処理は、時間的に後のフレームＦ３からの差分を演算し、その差分（ＳＰ２）を伝送するものである（後方予測符号化）。第３の処理は、時間的に先行するフレームＦ１との差分（ＳＰ３）を伝送するものである（前方予測符号化）。さらに第４の処理は、時間的に先行するフレームＦ１と後行するフレームＦ３の平均値との差分（ＳＰ４）を生成し、これを伝送データＦ２Ｘとして伝送するものである（両方向予測符号化）。

【００２３】実際には、上述した４つの方法のうちの伝送データが最も少なくなる方法が採用される。

【００２４】なお、差分データを伝送するとき、差分を演算する対象となるフレームの画像（予測画像）との間の動きベクトルｘ１（フレームＦ１とＦ２の間の動きベクトル）（前方予測の場合）、もしくはｘ２（フレームＦ３とＦ２の間の動きベクトル）（後方予測の場合）、またはｘ１とｘ２の両方（両方向予測の場合）が、差分データとともに伝送される。

【００２５】また、Ｐピクチャのフレーム Ｆ３は、時間的に先行するフレーム Ｆ１を予測画像として、このフレームとの差分信号（ＳＰ３）と、動きベクトルｘ３が演算され、これが伝送データ Ｆ３Ｘとして伝送される（前方予測符号化）。あるいはまた、元のフレーム Ｆ３のデータが、そのままデータ Ｆ３Ｘとして伝送される（ＳＰ１）（イントラ符号化）。いずれの方法により伝送されるかは、Ｂピクチャにおける場合と同様に、伝送データがより少なくなる方法が選択される。

【００２６】図５は、上述した原理に基づいて、動画像信号を符号化して伝送し、これを復号化する装置の構成例を示している。符号化装置１は、入力された映像信号を符号化し、伝送路としての記録媒体３に伝送するようになされている。そして、復号装置２は、記録媒体３に記録された信号を再生し、これを復号して出力するようになされている。

【００２７】符号化装置１においては、入力された映像信号が前処理回路１１に入力され、そこで輝度信号と色信号（本実施の形態の場合、色差信号）が分離され、それぞれA/D変換器１２，１３でアナログ信号がデジタル信号に変換される。A/D変換器１２，１３によりデジタル信号に変換された映像信号は、フレームメモリ１４に供給され、記憶される。フレームメモリ１４は、輝度信号を輝度信号フレームメモリ１５に、また、色差信号を色差信号フレームメモリ１６に、それぞれ記憶させる。

【００２８】フォーマット変換回路１７は、フレームメモリ１４に記憶されたフレームフォーマットの信号を、ブロックフォーマットの信号に変換する。すなわち、図６に示すように、フレームメモリ１４に記憶された映像信号は、１ライン当りＨドットのラインがＶライン集められた、図６(A)に示すようなフレームフォーマットのデータとされている。フォーマット変換回路１７は、この１フレームの信号を、図６(B)に示すように、１６ラインを単位としてＭ個のスライスに区分する。そして、各スライスは、Ｍ個のマクロブロックに分割される。マクロブロックは、図６(C)に示すように、１６×１６個の画素（ドット）に対応する輝度信号により構成され、この輝度信号は、さらに８×８ドットを単位とするブロックＹ［1］乃至Ｙ［4］に区分される。そして、この１６×１６ドットの輝度信号には、８×８ドットのＣｂ信号と、８×８ドットのＣｒ信号が対応される。

【００２９】このように、ブロックフォーマットに変換されたデータは、フォーマット変換回路１７からエンコーダ１８に供給され、ここでエンコード（符号化）が行われる。その詳細については、図７を参照して後述する。

【００３０】エンコーダ１８によりエンコードされた信号は、ビットストリームとして伝送路に出力される。例えば記録回路１９に供給され、デジタル信号として記録媒体３に記録される。

【００３１】再生回路３０により記録媒体３より再生されたデータは、復号装置２のデコーダ３１に供給され、デコードされる。デコーダ３１の詳細については、図１２を参照して後述する。

【００３２】デコーダ３１によりデコードされたデータは、フォーマット変換回路３２に入力され、ブロックフォーマットからフレームフォーマットに変換される。そして、フレームフォーマットの輝度信号は、フレームメモリ３３の輝度信号フレームメモリ３４に供給されて記憶され、色差信号は色差信号フレームメモリ３５に供給されて記憶される。輝度信号フレームメモリ３４と色差信号フレームメモリ３５から読み出された輝度信号と色差信号は、それぞれＤ／Ａ変換器３６，３７によりアナログ信号に変換され、後処理回路３８に供給される。後処理回路３８は、輝度信号と色差信号を合成して出力する。

【００３３】次に図７を参照して、エンコーダ１８の構成について説明する。符号化される画像データは、マクロブロック単位で動きベクトル検出回路５０に入力される。動きベクトル検出回路５０は、予め設定されている所定のシーケンスに従って、各フレームの画像データを、Ｉピクチャ、Ｐピクチャ、またはＢピクチャとして処理する。シーケンシャルに入力される各フレームの画像を、Ｉ，Ｐ、またはＢのいずれのピクチャとして処理するかは、予め定められている（例えば、図２と図３に示したように、フレーム Ｆ１乃至Ｆ１７により構成されるグループオブピクチャが、Ｉ，Ｂ，Ｐ，Ｂ，Ｐ，・・・Ｂ，Ｐとして処理される）。

【００３４】Ｉピクチャとして処理されるフレーム（例えば、フレーム Ｆ１）の画像データは、動きベクトル検出回路５０からフレームメモリ５１の前方原画像部５１ａに転送、記憶され、Ｂピクチャとして処理されるフレーム（例えば、フレーム Ｆ２）の画像データは、原画像部５１ｂに転送、記憶され、Ｐピクチャとして処理されるフレーム（例えば、フレーム Ｆ３）の画像データは、後方原画像部５１ｃに転送、記憶される。

【００３５】また、次のタイミングにおいて、さらにＢピクチャ（フレーム Ｆ４）またはＰピクチャ（フレーム Ｆ５）として処理すべきフレームの画像が入力されたとき、それまで後方原画像部５１ｃに記憶されていた最初のＰピクチャ（フレーム Ｆ３）の画像データが、前方原画像部５１ａに転送され、次のＢピクチャ（フレーム Ｆ４）の画像データが、原画像部５１ｂに記憶（上書き）され、次のＰピクチャ（フレーム Ｆ５）の画像データが、後方原画像部５１ｃに記憶（上書き）される。このような動作が順次繰り返される。

【００３６】フレームメモリ５１に記憶された各ピクチャの信号は、そこから読み出され、予測モード切り替え回路５２において、フレーム予測モード処理、またはフィールド予測モード処理が行なわれる。

【００３７】さらにまた、予測判定回路５４の制御の下に、演算部５３において、画像内予測、前方予測、後方予測、または両方向予測の演算が行なわれる。これらの処理のうち、いずれの処理を行なうかは、予測誤差信号（処理の対象とされている参照画像と、これに対する予測画像との差分）に対応して決定される。このため、動きベクトル検出回路５０は、この判定に用いられる予測誤差信号の絶対値和（自乗和でもよい）を生成する。

【００３８】ここで、予測モード切り替え回路５２におけるフレーム予測モードとフィールド予測モードについて説明する。

【００３９】フレーム予測モードが設定された場合においては、予測モード切り替え回路５２は、動きベクトル検出回路５０より供給される４個の輝度ブロックＹ［１］乃至Ｙ［４］を、そのまま後段の演算部５３に出力する。すなわち、この場合においては、図８に示すように、各輝度ブロックに奇数フィールドのラインのデータと、偶数フィールドのラインのデータとが混在した状態となっている。このフレーム予測モードにおいては、４個の輝度ブロック（マクロブロック）を単位として予測が行われ、４個の輝度ブロックに対して１個の動きベクトルが対応される。

【００４０】これに対して、予測モード切り替え回路５２は、フィールド予測モードにおいては、図８に示す構成で動きベクトル検出回路５０より入力される信号を、図９に示すように、４個の輝度ブロックのうち、輝度ブロックＹ［１］とＹ［２］を、例えば奇数フィールドのラインのドットだけで構成させ、他の２個の輝度ブロックＹ［３］とＹ［４］を、偶数フィールドのラインのドットだけで構成させて、演算部５３に出力する。この場合においては、２個の輝度ブロックＹ［１］とＹ［２］に対して、１個の動きベクトルが対応され、他の２個の輝度ブロックＹ［３］とＹ［４］に対して、他の１個の動きベクトルが対応される。

【００４１】動きベクトル検出回路５０は、フレーム予測モードにおける予測誤差の絶対値和、およびフィールド予測モードにおける予測誤差の絶対値和を予測モード切り替え回路５２に出力する。予測モード切り替え回路５２は、フレーム予測モードとフィールド予測モードにおける予測誤差の絶対値和を比較し、その値が小さい予測モードに対応する処理を施して、データを演算部５３に出力する。

【００４２】ただし、このような処理は、実際には動きベクトル検出回路５０で行われる。すなわち、動きベクトル検出回路５０は、決定されたモードに対応する構成の信号を予測モード切り替え回路５２に出力し、予測モード切り替え回路５２は、その信号を、そのまま後段の演算部５３に出力する。

【００４３】なお、色差信号は、フレーム予測モードの場合、図８に示すように、奇数フィールドのラインのデータと偶数フィールドのラインのデータとが混在する状態で、演算部５３に供給される。また、フィールド予測モードの場合、図９に示すように、各色差ブロックＣｂ，Ｃｒの上半分（４ライン）が、輝度ブロックＹ［１］，Ｙ［２］に対応する奇数フィールドの色差信号とされ、下半分（４ライン）が、輝度ブロックＹ［３］，Ｙ［４］に対応する偶数フィールドの色差信号とされる。

【００４４】また、動きベクトル検出回路５０は、以下に示すようにして、予測判定回路５４において、画像内予測、前方予測、後方予測、または両方向予測のいずれの予測を行なうかを決定するための予測誤差の絶対値和を生成する。

【００４５】すなわち、画像内予測の予測誤差の絶対値和として、参照画像のマクロブロックの信号Ａijの総和ΣＡijの絶対値｜ΣＡij｜と、マクロブロックの信号Ａijの絶対値｜Ａij｜の総和Σ｜Ａij｜の差を求める。また、前方予測の予測誤差の絶対値和として、参照画像のマクロブロックの信号Ａijと、予測画像のマクロブロックの信号Ｂijの差Ａij－Ｂijの絶対値｜Ａij－Ｂij｜の総和Σ｜Ａij－Ｂij｜を求める。また、後方予測と両方向予測の予測誤差の絶対値和も、前方予測における場合と同様に（その予測画像を前方予測における場合と異なる予測画像に変更して）求める。

【００４６】これらの絶対値和は、予測判定回路５４に供給される。予測判定回路５４は、前方予測、後方予測および両方向予測の予測誤差の絶対値和のうちの最も小さいものを、インタ予測の予測誤差の絶対値和として選択する。さらに、このインタ予測の予測誤差の絶対値和と、画像内予測の予測誤差の絶対値和とを比較し、その小さい方を選択し、この選択した絶対値和に対応するモードを予測モードとして選択する。すなわち、画像内予測の予測誤差の絶対値和の方が小さければ、画像内予測モードが設定される。インタ予測の予測誤差の絶対値和の方が小さければ、前方予測、後方予測または両方向予測モードのうちの対応する絶対値和が最も小さかったモードが設定される。

【００４７】このように、動きベクトル検出回路５０は、参照画像のマクロブロックの信号を、フレームまたはフィールド予測モードのうち、予測モード切り替え回路５２により選択されたモードに対応する構成で、予測モード切り替え回路５２を介して演算部５３に供給するとともに、４つの予測モードのうちの予測判定回路５４により選択された予測モードに対応する予測画像と参照画像の間の動きベクトルを検出し、可変長符号化回路５８と動き補償回路６４に出力する。上述したように、この動きベクトルとしては、対応する予測誤差の絶対値和が最小となるものが選択される。

【００４８】予測判定回路５４は、動きベクトル検出回路５０が前方原画像部５１ａよりＩピクチャの画像デー

タを読み出しているとき、予測モードとして、フレームまたはフィールド（画像）内予測モード（動き補償を行わないモード）を設定し、演算部５３のスイッチ５３ｄを接点ａ側に切り替える。これにより、Ｉピクチャの画像データがDCTモード切り替え回路５５に入力される。

【００４９】DCTモード切り替え回路５５は、図１０または図１１に示すように、４個の輝度ブロックのデータを、奇数フィールドのラインと偶数フィールドのラインが混在する状態（フレームDCTモード）、または、分離された状態（フィールドDCTモード）、のいずれかの状態にして、DCT回路５６に出力する。

【００５０】すなわち、DCTモード切り替え回路５５は、奇数フィールドと偶数フィールドのデータを混在してDCT処理した場合における符号化効率と、分離した状態においてDCT処理した場合の符号化効率とを比較し、符号化効率の良好なモードを選択する。

【００５１】例えば、入力された信号を、図１０に示すように、奇数フィールドと偶数フィールドのラインが混在する構成とし、上下に隣接する奇数フィールドのラインの信号と偶数フィールドのラインの信号の差を演算し、さらにその絶対値の和（または自乗和）を求める。

【００５２】また、入力された信号を、図１１に示すように、奇数フィールドと偶数フィールドのラインが分離した構成とし、上下に隣接する奇数フィールドのライン同士の信号の差と、偶数フィールドのライン同士の信号の差を演算し、それぞれの絶対値の和（または自乗和）を求める。

【００５３】さらに、両者（絶対値和）を比較し、小さい値に対応するDCTモードを設定する。すなわち、前者の方が小さければ、フレームDCTモードを設定し、後者の方が小さければ、フィールドDCTモードを設定する。

【００５４】そして、選択したDCTモードに対応する構成のデータをDCT回路５６に出力するとともに、選択したDCTモードを示すDCTフラグを、可変長符号化回路５８、および動き補償回路６４に出力する。

【００５５】予測モード切り替え回路５２における予測モード（図8と図9）と、このDCTモード切り替え回路５５におけるDCTモード（図１０と図１１）を比較して明らかなように、輝度ブロックに関しては、両者の各モードにおけるデータ構造は実質的に同一である。

【００５６】予測モード切り替え回路５２において、フレーム予測モード（奇数ラインと偶数ラインが混在するモード）が選択された場合、DCTモード切り替え回路５５においても、フレームDCTモード（奇数ラインと偶数ラインが混在するモード）が選択される可能性が高く、また予測モード切り替え回路５２において、フィールド予測モード（奇数フィールドと偶数フィールドのデータが分離されたモード）が選択された場合、DCTモード切り替え回路５５において、フィールドDCTモード（奇数フィールドと偶数フィールドのデータが分離されたモード）が選択される可能性が高い。

【００５７】しかしながら、必ずしも常にこのようにモードが選択されるわけではなく、予測モード切り替え回路５２においては、予測誤差の絶対値和が小さくなるようにモードが決定され、DCTモード切り替え回路５５においては、符号化効率が良好となるようにモードが決定される。

【００５８】DCTモード切り替え回路５５より出力されたＩピクチャの画像データは、DCT回路５６に入力されてDCT処理され、DCT係数に変換される。このDCT係数は、量子化回路５７に入力され、送信バッファ５９のデータ蓄積量（バッファ蓄積量）に対応した量子化スケールで量子化された後、可変長符号化回路５８に入力される。

【００５９】可変長符号化回路５８は、量子化回路５７より供給される量子化スケール（スケール）に対応して、量子化回路５７より供給される画像データ（いまの場合、Ｉピクチャのデータ）を、例えばハフマン符号などの可変長符号に変換し、送信バッファ５９に出力する。

【００６０】可変長符号化回路５８にはまた、量子化回路５７より量子化スケール（スケール）、予測判定回路５４より予測モード（画像内予測、前方予測、後方予測、または両方向予測のいずれが設定されたかを示すモード）、動きベクトル検出回路５０より動きベクトル、予測モード切り替え回路５２より予測フラグ（フレーム予測モードまたはフィールド予測モードのいずれが設定されたかを示すフラグ）、およびDCTモード切り替え回路５５が出力するDCTフラグ（フレームDCTモードまたはフィールドDCTモードのいずれが設定されたかを示すフラグ）が入力されており、これらも可変長符号化される。

【００６１】送信バッファ５９は、入力されたデータを一時蓄積し、蓄積量に対応するデータを量子化回路５７に出力する。送信バッファ５９は、そのデータ残量が許容上限値まで増量すると、量子化制御信号によって量子化回路５７の量子化スケールを大きくすることにより、量子化データのデータ量を低下させる。また、これとは逆に、データ残量が許容下限値まで減少すると、送信バッファ５９は、量子化制御信号によって量子化回路５７の量子化スケールを小さくすることにより、量子化データのデータ量を増大させる。このようにして、送信バッファ５９のオーバフローまたはアンダフローが防止される。

【００６２】そして、送信バッファ５９に蓄積されたデータは、所定のタイミングで読み出され、伝送路に出力され、例えば記録回路１９を介して記録媒体３に記録される。

【００６３】一方、量子化回路５７より出力されたＩピクチャのデータは、逆量子化回路６０に入力され、量子

化回路５７より供給される量子化スケールに対応して逆量子化される。逆量子化回路６０の出力は、IDCT（逆離散コサイン変換）回路６１に入力され、逆離散コサイン変換処理された後、演算器６２を介してフレームメモリ６３の前方予測画像部６３ａ供給されて記憶される。

【００６４】動きベクトル検出回路５０は、シーケンシャルに入力される各フレームの画像データを、たとえば、Ｉ，Ｂ，Ｐ，Ｂ，Ｐ，Ｂ・・・のピクチャとしてそれぞれ処理する場合、最初に入力されたフレームの画像データをＩピクチャとして処理した後、次に入力されたフレームの画像をＢピクチャとして処理する前に、さらにその次に入力されたフレームの画像データをＰピクチャとして処理する。Ｂピクチャは、後方予測を伴うため、後方予測画像としてのＰピクチャが先に用意されていないと、復号することができないからである。

【００６５】そこで動きベクトル検出回路５０は、Ｉピクチャの処理の次に、後方原画像部５１ｃに記憶されているＰピクチャの画像データの処理を開始する。そして、上述した場合と同様に、マクロブロック単位でのフレーム間差分（予測誤差）の絶対値和が、動きベクトル検出回路５０から予測モード切り替え回路５２と予測判定回路５４に供給される。予測モード切り替え回路５２と予測判定回路５４は、このＰピクチャのマクロブロックの予測誤差の絶対値和に対応して、フレーム／フィールド予測モード、または画像内予測、前方予測、後方予測、もしくは両方向予測の予測モードを設定する。

【００６６】演算部５３は、画像内予測モードが設定されたとき、スイッチ５３ｄを上述したように接点ａ側に切り替える。したがって、このデータは、Ｉピクチャのデータと同様に、DCTモード切り替え回路５５、DCT回路５６、量子化回路５７、可変長符号化回路５８、および送信バッファ５９を介して伝送路に伝送される。また、このデータは、逆量子化回路６０、IDCT回路６１、および演算器６２を介してフレームメモリ６３の後方予測画像部６３ｂに供給されて記憶される。

【００６７】また、前方予測モードが設定された場合、スイッチ５３ｄが接点ｂに切り替えられるとともに、フレームメモリ６３の前方予測画像部６３ａに記憶されている画像（いまの場合、Ｉピクチャの画像）データが読み出され、動き補償回路６４により、動きベクトル検出回路５０が出力する動きベクトルに対応して動き補償される。すなわち、動き補償回路６４は、予測判定回路５４より前方予測モードの設定が指令されたとき、前方予測画像部６３ａの読み出しアドレスを、動きベクトル検出回路５０が、現在、出力しているマクロブロックの位置に対応する位置から動きベクトルに対応する分だけずらしてデータを読み出し、予測画像データを生成する。

【００６８】動き補償回路６４より出力された予測画像データは、演算器５３ａに供給される。演算器５３ａは、予測モード切り替え回路５２より供給された参照画像のマクロブロックのデータから、動き補償回路６５より供給された、このマクロブロックに対応する予測画像データを減算し、その差分（予測誤差）を出力する。この差分データは、DCTモード切り替え回路５５、DCT回路５６、量子化回路５７、可変長符号化回路５８、および送信バッファ５９を介して伝送路に伝送される。また、この差分データは、逆量子化回路６０、およびIDCT回路６１により局所的に復号され、演算器６２に入力される。

【００６９】この演算器６２にはまた、演算器５３ａに供給されている予測画像データと同一のデータが供給されている。演算器６２は、IDCT回路６１が出力する差分データに、動き補償回路６４が出力する予測画像データを加算する。これにより、元の（復号した）Ｐピクチャの画像データが得られる。このＰピクチャの画像データは、フレームメモリ６３の後方予測画像部６３ｂに供給されて記憶される。

【００７０】動きベクトル検出回路５０は、このように、ＩピクチャとＰピクチャのデータが前方予測画像部６３ａと後方予測画像部６３ｂにそれぞれ記憶された後、次にＢピクチャの処理を実行する。予測モード切り替え回路５２と予測判定回路５４は、マクロブロック単位でのフレーム間差分の絶対値和の大きさに対応して、フレーム／フィールドモードを設定し、また、予測モードを画像内予測モード、前方予測モード、後方予測モード、または両方向予測モードのいずれかに設定する。

【００７１】上述したように、画像内予測モードまたは前方予測モードの時、スイッチ５３ｄは接点ａまたはｂに切り替えられる。このとき、Ｐピクチャにおける場合と同様の処理が行われ、データが伝送される。

【００７２】これに対して、後方予測モードまたは両方向予測モードが設定された時、スイッチ５３ｄは、接点ｃまたはｄにそれぞれ切り替えられる。

【００７３】スイッチ５３ｄが接点ｃに切り替えられている後方予測モードの時、後方予測画像部６３ｂに記憶されている画像（いまの場合、Ｐピクチャの画像）データが読み出され、動き補償回路６４により、動きベクトル検出回路５０が出力する動きベクトルに対応して動き補償される。すなわち、動き補償回路６４は、予測判定回路５４より後方予測モードの設定が指令されたとき、後方予測画像部６３ｂの読み出しアドレスを、動きベクトル検出回路５０が、現在、出力しているマクロブロックの位置に対応する位置から動きベクトルに対応する分だけずらしてデータを読み出し、予測画像データを生成する。

【００７４】動き補償回路６４より出力された予測画像データは、演算器５３ｂに供給される。演算器５３ｂは、予測モード切り替え回路５２より供給された参照画像のマクロブロックのデータから、動き補償回路６４より供給された予測画像データを減算し、その差分を出力

する。この差分データは、DCTモード切り替え回路５５、ＤＣＴ回路５６、量子化回路５７、可変長符号化回路５８、および送信バッファ５９を介して伝送路に伝送される。

【００７５】スイッチ５３ｄが接点ｄに切り替えられている両方向予測モードの時、前方予測画像部６３ａに記憶されている画像（いまの場合、Ｉピクチャの画像）データと、後方予測画像部６３ｂに記憶されている画像（いまの場合、Ｐピクチャの画像）データが読み出され、動き補償回路６４により、動きベクトル検出回路５０が出力する動きベクトルに対応して動き補償される。

【００７６】すなわち、動き補償回路６４は、予測判定回路５４より両方向予測モードの設定が指令されたとき、前方予測画像部６３ａと後方予測画像部６３ｂの読み出しアドレスを、動きベクトル検出回路５０がいま出力しているマクロブロックの位置に対応する位置から動きベクトル（この場合の動きベクトルは、前方予測画像用と後方予測画像用の２つとなる）に対応する分だけずらしてデータを読み出し、予測画像データを生成する。

【００７７】動き補償回路６４より出力された予測画像データは、演算器５３ｃに供給される。演算器５３ｃは、動きベクトル検出回路５０より供給された参照画像のマクロブロックのデータから、動き補償回路６４より供給された予測画像データの平均値を減算し、その差分を出力する。この差分データは、DCTモード切り替え回路５５、DCT回路５６、量子化回路５７、可変長符号化回路５８、および送信バッファ５９を介して伝送路に伝送される。

【００７８】Ｂピクチャの画像は、他の画像の予測画像とされることがないため、フレームメモリ６３には記憶されない。

【００７９】なお、フレームメモリ６３において、前方予測画像部６３ａと後方予測画像部６３ｂは、必要に応じてバンク切り替えが行われ、所定の参照画像に対して、一方または他方に記憶されているものを、前方予測画像あるいは後方予測画像として切り替えて出力することができる。

【００８０】上述した説明においては、輝度ブロックを中心として説明をしたが、色差ブロックについても同様に、図８乃至図１１に示すマクロブロックを単位として処理されて伝送される。なお、色差ブロックを処理する場合の動きベクトルは、対応する輝度ブロックの動きベクトルを垂直方向と水平方向に、それぞれ１／２にしたものが用いられる。

【００８１】図１２は、図５のデコーダ３１の構成を示すブロック図である。伝送路（記録媒体３）を介して伝送された符号化された画像データは、図示せぬ受信回路で受信されたり、再生装置で再生され、受信バッファ８１に一時記憶された後、復号回路９０の可変長復号化回路８２に供給される。可変長復号化回路８２は、受信バッファ８１より供給されたデータを可変長復号化し、動きベクトル、予測モード、予測フラグ、およびDCTフラグを動き補償回路８７に出力し、量子化スケールを逆量子化回路８３に出力するとともに、復号された画像データを逆量子化回路８３に出力する。

【００８２】逆量子化回路８３は、可変長復号化回路８２より供給された画像データを、同じく可変長復号化回路８２より供給された量子化スケールに従って逆量子化し、IDCT回路８４に出力する。逆量子化回路８３より出力されたデータ（DCT係数）は、IDCT回路８４により、逆離散コサイン変換処理が施され、演算器８５に供給される。

【００８３】IDCT回路８４より演算器８５に供給された画像データが、Ｉピクチャのデータである場合、そのデータは演算器８５より出力され、演算器８５に後に入力される画像データ（ＰまたはＢピクチャのデータ）の予測画像データ生成のために、フレームメモリ８６の前方予測画像部８６ａに供給されて記憶される。また、このデータは、フォーマット変換回路３２（図５）に出力される。

【００８４】IDCT回路８４より供給された画像データが、その１フレーム前の画像データを予測画像データとするＰピクチャのデータであり、前方予測モードのデータである場合、フレームメモリ８６の前方予測画像部８６ａに記憶されている、１フレーム前の画像データ（Ｉピクチャのデータ）が読み出され、動き補償回路８７で可変長復号化回路８２より出力された動きベクトルに対応する動き補償が施される。そして、演算器８５において、IDCT回路８４より供給された画像データ（差分のデータ）と加算され、出力される。この加算されたデータ、すなわち、復号されたＰピクチャのデータは、演算器８５に後に入力される画像データ（ＢピクチャまたはＰピクチャのデータ）の予測画像データ生成のために、フレームメモリ８６の後方予測画像部８６ｂに供給されて記憶される。

【００８５】Ｐピクチャのデータであっても、画像内予測モードのデータは、Ｉピクチャのデータと同様に、演算器８５において処理は行われず、そのまま後方予測画像部８６ｂに記憶される。

【００８６】このＰピクチャは、次のＢピクチャの次に表示されるべき画像であるため、この時点では、まだフォーマット変換回路３２へ出力されない（上述したように、Ｂピクチャの後に入力されたＰピクチャが、Ｂピクチャより先に処理され、伝送されている）。

【００８７】IDCT回路８４より供給された画像データが、Ｂピクチャのデータである場合、可変長復号化回路８２より供給された予測モードに対応して、フレームメモリ８６の前方予測画像部８６ａに記憶されているＩピクチャの画像データ（前方予測モードの場合）、後方予測画像部８６ｂに記憶されているＰピクチャの画像デー

タ（後方予測モードの場合）、または、その両方の画像データ（両方向予測モードの場合）が読み出され、動き補償回路８７において、可変長復号化回路８２より出力された動きベクトルに対応する動き補償が施されて、予測画像が生成される。但し、動き補償を必要としない場合（画像内予測モードの場合）、予測画像は生成されない。

【００８８】このようにして、動き補償回路８７で動き補償が施されたデータは、演算器８５において、IDCT回路８４の出力と加算される。この加算出力は、フォーマット変換回路３２に出力される。

【００８９】ただし、この加算出力はＢピクチャのデータであり、他の画像の予測画像生成のために利用されることがないため、フレームメモリ８６には記憶されない。

【００９０】Ｂピクチャの画像が出力された後、後方予測画像部８６ｂに記憶されているＰピクチャの画像データが読み出され、動き補償回路８７を介して演算器８５に供給される。但し、このとき、動き補償は行われない。

【００９１】なお、このデコーダ３１には、図５のエンコーダ１８における予測モード切り替え回路５２とＤＣＴモード切り替え回路５５に対応する回路が図示されていないが、これらの回路に対応する処理、すなわち、奇数フィールドと偶数フィールドのラインの信号が分離された構成を元の構成に必要に応じて戻す処理は、動き補償回路８７により実行される。

【００９２】また、上述した説明においては、輝度信号の処理について説明したが、色差信号の処理も同様に行われる。ただし、この場合の動きベクトルは、輝度信号用の動きベクトルを、垂直方向および水平方向に１／２にしたものが用いられる。

【００９３】図１３は、符号化された画像の品質を示している。画像の品質(SNR:Signal toNoise Ratio)は、ピクチャタイプに対応して制御され、Ｉピクチャ、およびＰピクチャは高品質とされ、Ｂピクチャは、Ｉ，Ｐピクチャに比べて劣る品質とされて伝送される。これは、人間の視覚特性を利用した手法であり、全ての画像品質を平均化するよりも、品質を振動させたほうが視覚上の画質が良くなるためである。このピクチャタイプに対応した画質の制御は、図７の量子化回路５７により実行される。

【００９４】図１４は、本発明を適用したトランスコーダ１０１の構成を示しており、図１５は、そのさらに詳細な構成を示している。復号装置１０２は、所定のビットレート（この例の場合、10Mbps）のビットストリームに含まれる（多重化されている）符号化された画像信号を、ビットストリームに含まれる（多重化されている）そのビットストリームの現符号化パラメータ（フレーム／フィールドDCTフラグ、フレーム／フィールド予測フラグ、予測モード、ピクチャタイプ、動きベクトル、マクロブロック情報、および量子化スケール）を用いて復号し、符号化パラメータ多重装置１０３に出力するとともに、現符号化パラメータも符号化パラメータ多重装置１０３に出力するようになされている。

【００９５】復号装置１０２はまた、ビットストリームに含まれるユーザデータを復号、分離し、履歴復号装置１０４に出力する。その詳細は後述するが、このユーザデータには、直近の３世代分の符号化パラメータで構成される世代履歴情報が含まれている。これに対して、現符号化パラメータは、例えばgroup_of_pictures_header(1), extension_and_user_data(1), picture_header(), picture_coding_extension(), extensions_data(2), picture_data()、または、sequence_extension()に含まれている（後述する図３８）。履歴復号装置１０４は、入力されたユーザデータを復号し、３世代分の符号化パラメータを含む世代履歴情報を符号化パラメータ多重装置１０３に出力する。

【００９６】なお、復号装置１０２は、図５の復号装置２のデコーダ３１（図１２）を図１６に示すデコーダ１１１に変更したものである。デコーダ１１１の可変長復号化回路１１２は、現符号化パラメータをビットストリームから抽出し、所定の回路に供給するとともに、世代履歴情報を含むユーザデータを抽出し、履歴復号装置１０４に出力するようになされている。デコーダ１１１のその他の構成は、デコーダ３１と同様であるので、その説明は省略する。

【００９７】符号化パラメータ多重装置１０３は、復号された画像データの空き領域（その詳細は、図１８を参照して説明する）に４世代分の符号化パラメータを書き込み（多重化し）、ベースバンドのデジタルビデオ信号として、粗結合された（符号化パラメータ伝送用の専用バス等が設けられていない）符号化パラメータ分離装置１０５に出力する。符号化パラメータ分離装置１０５は、ベースバンドのデジタルビデオ信号から、画像データと、符号化装置１０６で符号化に用いる符号化パラメータを分離して符号化装置１０６に供給するようになされている。

【００９８】符号化パラメータ分離装置１０５はまた、入力されたベースバンドのデジタルビデオ信号から、符号化装置１０６で用いる符号化パラメータを除く３世代分の符号化パラメータを抽出し、履歴符号化装置１０７に出力する。履歴符号化装置１０７は、入力された３世代分の符号化パラメータをユーザデータに書き込み、そのユーザデータを符号化装置１０６に出力する。

【００９９】符号パラメータが書き込まれる画像データのフォーマットについて、図１７と図１８を参照して説明する。１個のマクロブロックは、図１７に示すように、１６×１６画素で構成される。この１６×１６画素のデータは、８×８画素の輝度信号Y[0][x]乃至Y[4][x]

と、８×８画素の色差信号Cr[0][x]，Cr[1][x]およびCb[0][x]，Cb[1][x]（x＝２乃至９）から構成されている。例えば、輝度信号Y[0][9]は、８×８画素の１行目の画素（８画素）の輝度信号を示している。１画素当たりの輝度信号の情報量は８ビットなので、輝度信号Y[0][9]の情報量は、８（画素）×８（ビット）＝６４ビットとなる。色差信号についても同様である。

【０１００】これに対して、画像データのフォーマットは、図１８に示すように、１０行分の領域（Ｄ０乃至Ｄ９）が設けられているので、２行分の領域（Ｄ０，Ｄ１）が不要となる。この空き領域には、６４ビット×１６＝１０２４ビットの情報が記録できるので、この２行分の領域に本来の画像データ以外の符号化パラメータを書き込む。なお、１個のマクロブロックに対応する符号化パラメータは、２５６ビットの情報量があるので、この領域には、過去４回の符号化に使用された符号化パラメータを記録することができる。

【０１０１】符号化パラメータ多重装置１０３から符号化パラメータ分離装置１０５に伝送される画像データ（デジタルビデオ信号）には、輝度信号Ｙ、色差信号Ｃｒ，Ｃｂを記載する領域として、１０行分（Ｄ０乃至Ｄ９）の領域が設けられている。しかしながら実際に輝度信号Ｙ等が書き込まれる領域は、Ｄ２乃至Ｄ９の８行分の領域であり、Ｄ０，Ｄ１の領域は利用されない。そこで、この２ビットの領域を符号化パラメータの書き込み用領域として利用する。これにより、図１７の１６×１６画素の所定の位置の画素の下位２ビットに、符号化パラメータが書き込まれることとなる。

【０１０２】符号化装置１０６は、これから行う符号化のための符号化パラメータとして供給された現符号化パラメータを利用して画像データを符号化するとともに、履歴符号化装置１０７から供給されるユーザデータをビットストリームに多重化して、所定のビットレート（この例の場合、5Mbps）でSDTI(Serial Data Transfer Interface)１０８－ｉ（ｉ＝１，２，・・・，Ｎ）（後述する図３０）に出力するようになされている。

【０１０３】なお、符号化装置１０６は、図５の符号化装置１のエンコーダ１８（図７）を図１９に示すエンコーダ１２１に変更したものである。エンコーダ１２１は、エンコーダ１８から符号化パラメータを生成する動きベクトル検出回路５０、フレームメモリ５１、予測モード切り替え回路５２、予測判定回路５４、およびDCTモード切り替え回路５５を削除し、履歴符号化装置１０７の出力するユーザデータを可変長符号化回路５８で可変長符号化するようにしたものである。エンコーダ１２１のその他の構成は、エンコーダ１８と同様であるので、その説明は省略する。

【０１０４】次に、図１５における履歴復号装置１０４と履歴符号化装置１０７についてさらに説明する。同図に示すように、履歴復号装置１０４は、復号装置１０２より供給されるユーザデータをデコードするユーザデータデコーダ２０１、ユーザデータデコーダ２０１の出力を変換するコンバータ２０２、およびコンバータ２０２の出力から履歴情報を再生するヒストリデコーダ２０３により構成されている。

【０１０５】また、履歴符号化装置１０７は、符号化パラメータ分離装置１０５より供給される３世代分の符号化パラメータをフォーマット化するヒストリフォーマッタ２１１、ヒストリフォーマッタ２１１の出力を変換するコンバータ２１２、コンバータ２１２の出力をユーザデータのフォーマットにフォーマットするユーザデータフォーマッタ２１３により構成されている。

【０１０６】ユーザデータデコーダ２０１は、復号装置１０２より供給されるユーザデータをデコードして、コンバータ２０２に出力する。詳細は後述するが、ユーザデータ（user_data()）は、user_data_start_codeとuser_dataからなり、MPEG規格においてはuser_dataの中に、連続する２３ビットの”０”を発生させることを禁止している。これは、start_codeを誤検出されないようにするためである。履歴情報内には、このような連続する２３ビット以上の”０”が存在することがあり得るので、これを処理して、converted_history_stream()（後述する図３８）に変換する必要がある。この変換を行うのは、履歴符号化装置１０７のコンバータ２１２である。履歴復号装置１０４のコンバータ２０２は、このコンバータ２１２と逆の変換処理を行うものである。

【０１０７】ヒストリデコーダ２０３は、コンバータ２０２の出力から履歴情報を生成し、符号化パラメータ多重装置１０３に出力する。

【０１０８】一方、履歴符号化装置１０７においては、ヒストリフォーマッタ２１１が符号化パラメータ分離装置１０５より供給される３世代分の符号化パラメータを履歴情報のフォーマットに変換する。このフォーマットには、固定長のもの（後述する図４０乃至図４６）と、可変長のもの（後述する図４７）とがある。これらの詳細については後述する。

【０１０９】ヒストリフォーマッタ２１１により、フォーマット化された履歴情報は、コンバータ２１２において、converted_history_stream()に変換される。これは、上述したように、user_data()のstart_codeが誤検出されないようにするためのものである。すなわち、履歴情報内には連続する２３ビット以上の”０”が存在するが、user_data中には連続する２３ビット以上の”０”を配置することができないので、この禁止項目に触れないようにコンバータ２１２によりデータを変換するのである。

【０１１０】ユーザデータフォーマッタ２１３は、コンバータ２１２より供給されるconverted_history_stream()に、後述する図３８に基づいて、Data_IDを付加し、さらに、user_data_stream_codeを付加して、video str

eam中に挿入できるuser_dataを生成し、符号化装置１０６に出力する。

【０１１１】図２０は、ヒストリフォーマッタ２１１の構成例を表している。その符号語変換器３０１と符号長変換器３０５には、符号化パラメータ（今回、履歴情報として伝送する符号化パラメータ）（項目データ）と、この符号化パラメータを配置するストリームを特定する情報（例えば、シンタックスの名称）（例えば、後述するsequence_headerの名称）（項目NO.）が、符号化パラメータ分離装置１０５から供給されている。符号語変換器３０１は、入力された符号化パラメータを、指示されたシンタックスに対応する符号語に変換し、バレルシフタ３０２に出力する。バレルシフタ３０２は、符号語変換器３０１より入力された符号語を、アドレス発生回路３０６より供給されるシフト量に対応する分だけシフトし、バイト単位の符号語として、スイッチ３０３に出力する。アドレス発生回路３０６が出力するビットセレクト信号により切り換えられるスイッチ３０３は、ビット分設けられており、バレルシフタ３０２より供給される符号語を、RAM３０４に供給し、記憶させる。このときの書き込みアドレスは、アドレス発生回路３０６から指定される。また、アドレス発生回路３０６から読み出しアドレスが指定されたとき、RAM３０４に記憶されているデータ（符号語）が読み出され、後段のコンバータ２１２に供給されるとともに、必要に応じて、スイッチ３０３を介してRAM３０４に再び供給され、記憶される。

【０１１２】符号長変換器３０５は、入力されるシンタックスと符号化パラメータとから、その符号化パラメータの符号長を決定し、アドレス発生回路３０６に出力する。アドレス発生回路３０６は、入力された符号長に対応して、上述したシフト量、ビットセレクト信号、書き込みアドレス、または読み出しアドレスを生成し、それらを、それぞれバレルシフタ３０２、スイッチ３０３、またはRAM３０４に供給する。

【０１１３】以上のように、ヒストリフォーマッタ２１１は、いわゆる可変長符号化器として構成され、入力された符号化パラメータを可変長符号化して出力する。

【０１１４】図２１は、以上のようにしてヒストリフォーマット化されたデータをデコードするヒストリデコーダ２０３の構成例を表している。このヒストリデコーダ２０３には、コンバータ２０２から供給された符号化パラメータのデータがRAM３１１に供給されて、記憶される。このときの書き込みアドレスは、アドレス発生回路３１５から供給される。アドレス発生回路３１５はまた、所定のタイミングで読み出しアドレスを発生し、RAM３１１に供給する。このとき、RAM３１１は、読み出しアドレスに記憶されているデータを読み出し、バレルシフタ３１２に出力する。バレルシフタ３１２は、アドレス発生回路３１５が出力するシフト量に対応する分だけ、入力されるデータをシフトし、逆符号長変換器３１３と逆符号語変換器３１４に出力する。

【０１１５】逆符号長変換器３１３にはまた、コンバータ２０２から、符号化パラメータが配置されているストリームのシンタックスの名称が供給されている。逆符号長変換器３１３は、そのシンタックスに基づいて、入力されたデータ（符号語）から符号長を求め、求めた符号長をアドレス発生回路３１５に出力する。

【０１１６】また、逆符号語変換器３１４は、バレルシフタ３１２より供給されたデータを、シンタックスに基づいて復号し（逆符号語化し）、符号化パラメータ多重装置１０３に出力する。

【０１１７】また、逆符号語変換器３１４は、どのような符号語が含まれているのかを特定するのに必要な情報（符号語の区切りを決定するのに必要な情報）を抽出し、アドレス発生回路３１５に出力する。アドレス発生回路３１５は、この情報と逆符号長変換器３１３より入力された符号長に基づいて、書き込みアドレスおよび読み出しアドレスを発生し、RAM３１１に出力するとともに、シフト量を発生し、バレルシフタ３１２に出力する。

【０１１８】図２２は、コンバータ２１２の構成例を表している。この例においては、ヒストリフォーマッタ２１１とコンバータ２１２の間に配置されているバッファメモリ３２０の、コントローラ３２６が出力する読み出しアドレスから８ビットのデータが読み出され、D型フリップフロップ（Ｄ－ＦＦ）３２１に供給され、保持されるようになされている。そして、D型フリップフロップ３２１より読み出されたデータは、スタッフ回路３２３に供給されるとともに、８ビットのD型フリップフロップ３２２にも供給され、保持される。D型フリップフロップ３２２より読み出された８ビットのデータは、D型フリップフロップ３２１より読み出された８ビットのデータと合成され、１６ビットのパラレルデータとして、スタッフ回路３２３に供給される。

【０１１９】スタッフ回路３２３は、コントローラ３２６より供給されるスタッフ位置を示す信号（stuff position）の位置に符号″１″を挿入し（スタッフィングし）、合計１７ビットのデータとして、バレルシフタ３２４に出力する。

【０１２０】バレルシフタ３２４は、コントローラ３２６より供給されるシフト量を示す信号（shift）に基づいて入力されたデータをシフトして、８ビットのデータを抽出し、８ビットのD型フリップフロップ３２５に出力する。D型フリップフロップ３２５に保持されたデータは、そこから読み出され、バッファメモリ３２７を介して、後段のユーザデータフォーマッタ２１３に供給される。この時、コントローラ３２６は、出力するデータとともに、書き込みアドレスを発生し、コンバータ２１２とユーザデータフォーマッタ２１３との間に介在するバッファメモリ３２７に供給する。

【０１２１】図２３は、スタッフ回路３２３の構成例を表している。D型フリップフロップ３２２，３２１より入力された１６ビットのデータは、それぞれスイッチ３３１－１６乃至３３１－１の接点ａに入力されている。スイッチ３３１－ｉ（ｉ＝０乃至１５）の接点ｃには、MSB側（図中上方）に隣接するスイッチのデータが供給されている。例えば、スイッチ３３１－１２の接点ｃには、MSB側に隣接するスイッチ３３１－１３の接点ａに供給されているLSBから１３番目のデータが供給されており、スイッチ３３１－１３の接点ｃには、MSB側に隣接するスイッチ３３１－１３の接点ａに供給されているLSB側から１４番目のデータが供給されている。

【０１２２】但し、LSBに対応するスイッチ３３１－１よりさらに下側のスイッチ３３１－０の接点ａは、開放されている。また、MSBに対応するスイッチ３３１－１６の接点ｃは、それより上位のスイッチが存在しないため、開放されている。

【０１２３】各スイッチ３３１－０乃至３３１－１６の接点ｂには、データ″１″が供給されている。

【０１２４】デコーダ３３２は、コントローラ３２６より供給されるデータ″１″を挿入する位置を示す信号stuff positionに対応して、スイッチ３３１－０乃至３３１－１６のうち、１つのスイッチを接点ｂ側に切り替え、それよりLSB側のスイッチは、接点ｃ側にそれぞれ切り替え、それよりMSB側のスイッチは、接点ａ側に切り替えさせる。

【０１２５】図２３は、LSB側から１３番目にデータ″１″を挿入する場合の例を示している。従って、この場合、スイッチ３３１－０乃至スイッチ３３１－１２は、いずれも接点ｃ側に切り替えられ、スイッチ３３１－１３は、接点ｂ側に切り替えられ、スイッチ３３１－１４乃至スイッチ３３１－１６は、接点ａ側に切り替えられている。

【０１２６】図２２のコンバータ２１２は、以上のような構成により、２２ビットの符号を２３ビットに変換して、出力することになる。

【０１２７】図２４は、図２２のコンバータ２１２の各部の出力データのタイミングを表している。コンバータ２１２のコントローラ３２６がバイト単位のクロックに同期して、読み出しアドレス（図２４（Ａ））を発生すると、バッファメモリ３２０から、それに対応するデータが、バイト単位で読み出され、D型フリップフロップ３２１に一旦保持される。そして、D型フリップフロップ３２１より読み出されたデータ（図２４（Ｂ））は、スタッフ回路３２３に供給されるとともに、D型フリップフロップ３２２に供給され、保持される。D型フリップフロップ３２２に保持されたデータは、そこからさらに読み出され（図２４（Ｃ））、スタッフ回路３２３に供給される。

【０１２８】従って、スタッフ回路３２３の入力（図２４（Ｄ））は、読み出しアドレスＡ１のタイミングにおいて、最初の１バイトのデータＤ０とされ、次の読み出しアドレスＡ２のタイミングにおいて、１バイトのデータＤ０と１バイトのデータＤ１より構成される２バイトのデータとなり、さらに読み出しアドレスＡ３のタイミングにおいては、データＤ１とデータＤ２より構成される２バイトのデータとなる。

【０１２９】スタッフ回路３２３には、データ″１″を挿入する位置を示す信号stuff position（図２４（Ｅ））がコントローラ３２６より供給される。スタッフ回路３２３のデコーダ３３２は、スイッチ３３１－１６乃至３３１－０のうち、この信号stuff positionに対応するスイッチを接点ｂに切り換え、それよりLSB側のスイッチを接点ｃ側に切り換え、さらにそれよりMSB側のスイッチを接点ａ側に切り換える。これにより、データ″１″が挿入されるので、スタッフ回路３２３からは、信号stuff positionで示す位置に、データ″１″が挿入されたデータ（図２４（Ｆ））が出力される。

【０１３０】バレルシフタ３２４は、入力されたデータを、コントローラ３２６より供給される信号shift（図２４（Ｇ））で示される量だけバレルシフトして、出力する（図２４（Ｈ））。この出力がさらにD型フリップフロップ３２５で一旦保持された後、後段に出力される（図２４（Ｉ））。

【０１３１】D型フリップフロップ３２５より出力されるデータには、２２ビットのデータの次に、データ″１″が挿入されている。従って、データ″１″と、次のデータ″１″の間には、その間のビットが全て０であったとしても、０のデータの連続する数は２２となる。

【０１３２】図２５は、コンバータ２０２の構成例を表している。このコンバータ２０２のD型フリップフロップ３４１乃至コントローラ３４６よりなる構成は、図２２に示したコンバータ２１２のD型フリップフロップ３２１乃至コントローラ３２６と基本的に同様の構成であるが、コンバータ２１２におけるスタッフ回路３２３に代えて、ディリート回路３４３が挿入されている点がコンバータ２１２におけり場合と異なっている。その他の構成は、図２２のコンバータ２１２における場合と同様である。

【０１３３】すなわち、このコンバータ２０２においては、コントローラ３４６が出力する削除するビットの位置を示す信号delete positionに従って、ディリート回路３４３が、そのビット（図２２のスタッフ回路３２３で挿入されたデータ″１″）が削除される。

【０１３４】その他の動作は、図２２のコンバータ２１２における場合と同様である。

【０１３５】図２６は、ディリート回路３４３の構成例を表している。この構成例においては、D型フリップフロップ３４２，３４１より入力された１６ビットのデータのうち、LSB側の１５ビットが、それぞれ対応するス

イッチ３５１－０乃至３５１－１４の接点ａに供給されている。各スイッチの接点ｂには、１ビットだけMSB側のデータが供給されている。デコーダ３５２は、コントローラ３４６より供給される信号delete positionにより指定されるビットを削除して、１５ビットのデータとして出力するようになされている。

【０１３６】図２６は、LSBから第１３番目のビットがディリートされる状態を示している。従って、この場合、スイッチ３５１－０乃至スイッチ３５１－１１が接点ａ側に切り替えられ、LSBから第１２番目までの１２ビットが、そのまま選択、出力されている。また、スイッチ３５１－１２乃至３５１－１４は、それぞれ接点ｂ側に切り替えられているので、第１４番目乃至第１６番目のデータが、第１３番目乃至第１５番目のビットのデータとして選択、出力される。

【０１３７】図２３のスタッフ回路３２３および図２６のディリート回路３４３の入力が１６ビットとなっているのは、それぞれ図２２のコンバータ２１２のスタッフ回路３２３の入力が、D型フリップフロップ３２２，３２１より供給される１６ビットとされており、また、図２５のコンバータ２０２においても、ディリート回路３４３の入力が、D型フリップフロップ３４２，３４１により１６ビットとされているためである。図２２において、スタッフ回路３２３の出力する１７ビットをバレルシフタ３２４でバレルシフトすることにより、例えば８ビットを最終的に選択、出力しているのと同様に、図２５のコンバータ２０２においても、ディリート回路３４３の出力する１５ビットのデータを、バレルシフタ３４４で所定量だけバレルシフトすることにより、８ビットのデータとしている。

【０１３８】図２７は、コンバータ２１２の他の構成例を表している。この構成例においては、カウンタ３６１が入力データのうち、連続する０のビットの数をカウントし、そのカウント結果をコントローラ３２６に出力するようになされている。コントローラ３２６は、例えばカウンタ３６１が連続する０のビットを２２個カウントしたとき、信号stuff positionをスタッフ回路３２３に出力する。また、このとき、コントローラ３２６は、カウンタ３６１をリセットし、再び連続する０のビットの数をカウンタ３６１にカウントさせる。

【０１３９】その他の構成と動作は、図２２における場合と同様である。

【０１４０】図２８は、コンバータ２０２の他の構成例を表している。この構成例においては、入力データのうち、連続する０の数をカウンタ３７１がカウントし、そのカウント結果をコントローラ３４６に出力するようになされている。カウンタ３７１のカウント値が２２に達したとき、コントローラ３４６は、信号delete positionをディリート回路３４３に出力するとともに、カウンタ３７１をリセットし、再び新たな連続する０のビットの数をカウンタ３７１にカウントさせる。その他の構成は、図２５における場合と同様である。

【０１４１】このように、この構成例においては、所定のパターン（データ″０″の連続する数）に基づいて、マーカービットとしてのデータ″１″が挿入され、また、削除されることになる。

【０１４２】図２７と図２８に示す構成は、図２２と図２５に示す構成よりも効率的な処理が可能となる。但し、変換後の長さが元の履歴情報に依存することになる。

【０１４３】図２９は、ユーザデータフォーマッタ２１３の構成例を表している。この例においては、コントローラ３８３がコンバータ２１２とユーザデータフォーマッタ２１３との間に配置されているバッファメモリ（図示せず）に読み出しアドレスを出力すると、そこから読み出されたデータが、ユーザデータフォーマッタ２１３のスイッチ３８２の接点ａ側に供給される。ROM３８１には、ユーザデータスタートコード、データＩＤなどのuser_data()を生成するのに必要なデータが記憶されている。コントローラ３１３は、所定のタイミングにおいて、スイッチ３８２を接点ａ側または接点ｂ側に切り替え、ROM３８１に記憶されているデータ、またはコンバータ２１２より供給されるデータを適宜選択し、出力する。これにより、user_data()のフォーマットのデータが符号化装置１０６に出力される。

【０１４４】なお、図示は省略するが、ユーザデータデコーダ２０１は、図２９のROM３８１より読み出され、挿入されたデータを削除するスイッチを介して、入力データを出力するようにすることで実現することができる。

【０１４５】図３０は、例えば映像編集スタジオにおいて、複数のトランスコーダ１０１－１乃至１０１－Nが直列に接続されて使用される状態を示している。各トランスコーダ１０１－ｉ（ｉ＝１乃至N）の符号化パラメータ多重装置１０３－ｉは、上述した符号化パラメータ用の領域の最も古い符号化パラメータが記録されている区画に、自己が用いた最新の符号化パラメータを上書きする。このことにより、ベースバンドの画像データには、同一のマクロブロックに対応する直近の４世代分の符号化パラメータ（世代履歴情報）が記録されることになる。

【０１４６】各符号化装置１０６－ｉのエンコーダ１２１－ｉ（図１９）は、その可変長符号化回路５８において、符号化パラメータ分離装置１０５－ｉから供給される今回用いる符号化パラメータに基づいて、量子化回路５７より供給されるビデオデータを符号化する。このようにして生成されるビットストリーム（例えば、picture_header()）中に、その現符号化パラメータは多重化される。

【０１４７】可変長符号化回路５８はまた、履歴符号化

装置１０７－ｉより供給されるユーザデータ（世代履歴情報を含む）を、出力するビットストリーム中に多重化する（図１８に示すような埋め込み処理ではなく、ビットストリーム中に多重化される）。そして、符号化装置１０６－ｉの出力するビットストリームは、SDTI１０８－ｉを介して、後段のトランスコーダ１０１－（ｉ＋１）に入力される。

【０１４８】トランスコーダ１０１－ｉとトランスコーダ１０１－（ｉ＋１）は、それぞれ図１５に示すように構成されている。従って、その処理は、図１５を参照して説明した場合と同様となる。

【０１４９】実際の符号化パラメータの履歴を利用した符号化として、現在Ｉピクチャとして符号化されていたものを、ＰもしくはＢピクチャに変更したい場合、過去の符号化パラメータの履歴を見て、過去にＰもしくはＢピクチャであった場合を探し、これらの履歴が存在した場合は、その動きベクトルなどのパラメータを利用して、ピクチャタイプを変更する。反対に過去に履歴がない場合は、動き検出を行わないピクチャタイプの変更を断念する。もちろん履歴がない場合であっても、動き検出を行えばピクチャタイプを変更できる。

【０１５０】図１８に示すフォーマットの場合、４世代分の符号化パラメータを埋め込むようにしたが、Ｉ、Ｐ、Ｂの各ピクチャタイプのパラメータを埋め込むようにすることもできる。図３１は、この場合のフォーマットの例を示している。この例では、同一のマクロブロックが、過去にピクチャタイプの変更を伴って符号化されたときにおける、ピクチャタイプ毎に１世代分の符号化パラメータ（ピクチャ履歴情報）が記録される。したがって、図１６に示したデコーダ１１１、および図１９に示したエンコーダ１２１は、現在（最新）、１世代前、２世代前、および３世代前の符号化パラメータの代わりに、Ｉピクチャ、Ｐピクチャ、およびＢピクチャに対応する１世代分の符号化パラメータを入出力することになる。

【０１５１】また、この例の場合、Cb[1][x]とCr[1][x]の領域は利用しないので、Cb[1][x]とCr[1][x]の領域を有さない４：２：０フォーマットの画像データにも本発明を適用することができる。

【０１５２】この例の場合、復号装置１０２は、符号化パラメータを復号と同時に取り出し、ピクチャタイプを判定して、画像信号のピクチャタイプに対応した場所に符号化パラメータを書き込んで（多重化して）符号化パラメータ分離装置１０５に出力する。符号化パラメータ分離装置１０５は、符号化パラメータを分離し、これから符号化したいピクチャタイプと、入力された過去の符号化パラメータを考慮して、ピクチャタイプを変更しながら再符号化を行うことができる。

【０１５３】次に、各トランスコーダ１０１において、変更が可能なピクチャタイプを判定する処理について、図３２のフローチャートを参照して説明する。なお、この処理はトランスコーダ１０１におけるピクチャタイプの変更は、過去の動きベクトルを利用するので、動き検出を行わないで実行されることを前提としている。また、以下に説明する処理は、符号化パラメータ分離装置１０５により実行される。

【０１５４】ステップＳ１において、ピクチャタイプ毎に１世代分の符号化パラメータ（ピクチャ履歴情報）が符号化パラメータコントローラ１２２に入力される。

【０１５５】ステップＳ２において、符号化パラメータ分離装置１０５は、ピクチャ履歴情報にＢピクチャに変更したときの符号化パラメータが存在するか否かを判定する。ピクチャ履歴情報にＢピクチャに変更したときの符号化パラメータが存在すると判定された場合、ステップＳ３に進む。

【０１５６】ステップＳ３において、符号化パラメータ分離装置１０５は、ピクチャ履歴情報にＰピクチャに変更したときの符号化パラメータが存在するか否かを判定する。ピクチャ履歴情報にＰピクチャに変更したときの符号化パラメータが存在すると判定された場合、ステップＳ４に進む。

【０１５７】ステップＳ４において、符号化パラメータ分離装置１０５は、変更可能なピクチャタイプがＩピクチャ、Ｐピクチャ、およびＢピクチャであると判断する。

【０１５８】ステップＳ３において、ピクチャ履歴情報にＰピクチャに変更したときの符号化パラメータが存在しないと判定された場合、ステップＳ５に進む。

【０１５９】ステップＳ５において、符号化パラメータ分離装置１０５は、変更可能なピクチャタイプがＩピクチャ、およびＢピクチャであると判断する。さらに、符号化パラメータ分離装置１０５は、特殊処理（Ｂピクチャの履歴情報に含まれる後方予測ベクトルを使わず、前方予測ベクトルだけを使う）を施すことにより、擬似的にＰピクチャに変更可能であると判断する。

【０１６０】ステップＳ２において、ピクチャ履歴情報にＢピクチャに変更したときの符号化パラメータが存在しないと判定された場合、ステップＳ６に進む。

【０１６１】ステップＳ６において、符号化パラメータ分離装置１０５は、ピクチャ履歴情報にＰピクチャに変更したときの符号化パラメータが存在するか否かを判定する。ピクチャ履歴情報にＰピクチャに変更したときの符号化パラメータが存在すると判定された場合、ステップＳ７に進む。

【０１６２】ステップＳ７において、符号化パラメータ分離装置１０５は、変更可能なピクチャタイプがＩピクチャ、およびＰピクチャであると判断する。さらに、符号化パラメータ分離装置１０５は、特殊処理（Ｐピクチャに履歴情報に含まれる前方予測ベクトルだけを使う）を施すことにより、Ｂピクチャに変更可能であると判断

する。

【０１６３】ステップＳ６において、ピクチャ履歴情報にＰピクチャに変更したときの符号化パラメータが存在しないと判定された場合、ステップＳ８に進む。ステップＳ８において、符号化パラメータ分離装置１０５は、動きベクトルが存在しないので、変更可能なピクチャタイプがＩピクチャだけである（Ｉピクチャなのでピクチャ以外には変更できない）と判断する。

【０１６４】ステップＳ４，Ｓ５，Ｓ７，Ｓ８の処理の次にステップＳ９において、符号化パラメータ分離装置１０５は、変更可能なピクチャタイプを表示装置（図示せず）に表示してユーザに通知する。

【０１６５】図３３は、ピクチャタイプ変更の例を示している。ピクチャタイプの変更は、GOPを構成するフレーム数が変更される。すなわち、この例の場合、N=15（GOPのフレーム数N=15）、M=3（GOP内のＩ、またはＰピクチャの出現周期M=3)のフレームから構成されるLong GOP（第１世代）から、N=1，M=1のフレームで構成されるShort GOP（第２世代）に変換され、再度、N=15，M=3のフレームから構成されるLong GOP（第３世代）に変換されている。なお、図中において破線は、GOPの境界を示している。

【０１６６】第１世代から第２世代にピクチャタイプが変更される場合において、上述した変更可能ピクチャタイプ判定処理の説明から明らかなように、全てのフレームは、ピクチャタイプをＩピクチャに変更することが可能である。このピクチャタイプ変更のとき、動画像（第０世代）が第１世代に変換されたときに演算された全ての動きベクトルは、ピクチャ履歴情報に保存された（残された）状態となる。次に、再度Long GOPに変換される（第２世代から第３世代にピクチャタイプが変更される）場合、第０世代から第１世代に変換されたときのピクチャタイプ毎の動きベクトルが保存されているので、これを再利用することにより、画質劣化を抑えて、再度、Long GOPに変換することが可能となる。

【０１６７】図３４は、ピクチャタイプ変更の他の例を示している。この例の場合、N=14，M=2であるLong GOP（第１世代）から、N=2，M=2であるShort GOP（第２世代）に変換され、さらに、N=１，M=1であるフレーム数が１のShort GOP（第３世代）に変換されて、フレーム数Nがランダムな GOP（第４世代）に変換される。

【０１６８】この例においても、第０世代から第１世代に変換されたときのピクチャタイプ毎の動きベクトルが、第３世代から第４世代への変換のときまで保存される。そこで、図３４に示すように、複雑にピクチャタイプを変更しても、保存されている符号化パラメータを再利用されることにより、画質劣化を小さく抑えることができる。さらに、保存されている符号化パラメータの量子化スケールを有効に利用すれば画質劣化の少ない符号化を実現できる。

【０１６９】この量子化スケールの再利用について、図３５を参照して説明する。図３５は、所定のフレームが、第１世代から第４世代まで常に、Ｉピクチャに変換されており、ビットレートだけが、４Mbps，１８Mbps、または５０Mbpsに変更されていることを示している。

【０１７０】例えば、第１世代(４Mbps)から第２世代(１８Mbps)への変換の際に、ビットレートの高速化に伴って、細かい量子化スケールで再符号化しても画質は向上しない。なぜならば、過去において粗い量子化ステップで量子化されたデータは、復元しないからである。したがって、図３５に示すように、途中でビットレートが高速化しても、それに伴って細かい量子化ステップで量子化することは、情報量が増加するだけであって画質の向上には繋がらない。したがって、過去のもっとも粗い（大きい）量子化スケールを維持するように制御すれば、最も無駄が無く、効率的な符号化が可能となる。

【０１７１】上述したように、ビットレートが変更されるときは、過去の量子化スケールの履歴を利用して符号化することは非常に有効である。

【０１７２】この量子化制御処理について、図３６のフローチャートを参照して説明する。ステップＳ１１において、符号化パラメータ分離装置１０５は、入力されたピクチャ履歴情報に、いまから変換するピクチャタイプの符号化パラメータが存在するか否かを判定する。変換するピクチャタイプの符号化パラメータが存在すると判定された場合、ステップＳ１２に進む。

【０１７３】ステップＳ１２において、符号化パラメータ分離装置１０５は、ピクチャ履歴情報の対照となる符号化パラメータから量子化スケール(Q_history)を抽出する。

【０１７４】ステップＳ１３において、符号化パラメータ分離装置１０５は、送信バッファ５９から量子化回路５７にフィードバックされる量子化スケールの候補値Q_feedbackを読み取る。

【０１７５】ステップＳ１４において、符号化パラメータ分離装置１０５は、Q_historyがQ_feedbackよりも大きい（粗い）か否かを判定する。Q_historyがQ_feedbackよりも大きいと判定された場合、ステップＳ１５に進む。

【０１７６】ステップＳ１５において、符号化パラメータ分離装置１０５は、量子化スケールとしてQ_historyを量子化回路５７に出力する。量子化回路５７は、Q_historyを用いて量子化を実行する。

【０１７７】ステップＳ１６において、フレームに含まれる全てのマクロブロックが量子化されたか否かが判定される。全てのマクロブロックが量子化されていないと判定された場合、ステップＳ１３に戻り、ステップＳ１３乃至Ｓ１６の処理が、全てのマクロブロックが量子化されるまで繰り返される。

【０１７８】ステップＳ１４において、Q_historyがQ_f

eedbackよりも大きくない（細かい）いと判定された場合、ステップＳ１７に進む。

【０１７９】ステップＳ１７において、符号化パラメータ分離装置１０５は、量子化スケールとしてQ_feedbackを量子化回路５７に出力する。量子化回路５７は、Q_feedbackを用いて量子化を実行する。

【０１８０】ステップＳ１１において、変換するピクチャタイプの符号化パラメータが存在しないと判定された場合、ステップＳ１８に進む。

【０１８１】ステップＳ１８において、量子化回路５７は、送信バッファ５９からフィードバックされる量子化スケールの候補値Q_feedbackを受け付ける。

【０１８２】ステップＳ１９において、量子化回路５７は、Q_feedbackを用いて量子化を実行する。

【０１８３】ステップＳ２０において、フレームに含まれる全てのマクロブロックが量子化されたか否かが判定される。全てのマクロブロックが量子化されていないと判定された場合、ステップＳ１８に戻り、ステップＳ１８乃至Ｓ２０の処理が、全てのマクロブロックが量子化されるまで繰り返される。

【０１８４】なお、本実施の形態におけるトランスコーダ１０１の内部においては、上述したように、復号側と符号側が粗結合されており、符号化パラメータを画像データに多重化させて伝送させたが、図３７に示すように、復号装置１０２と符号化装置１０６を符号化パラメータ伝送用の高速バス１１１で接続する（密結合する）ようにしてもよい。

【０１８５】図３８は、ＭＰＥＧのビデオストリームをデコードするためのシンタックスを表わした図である。デコーダは、このシンタックスに従ってＭＰＥＧビットストリームをデコードすることによって、ビットストリームから意味のある複数のデータ項目（データエレメント）を抽出する。以下に説明するシンタックスは、図において、その関数や条件文は細活字で表わされ、そのデータエレメントは、太活字で表されている。データ項目は、その名称、ビット長、及びそのタイプと伝送順序を示すニーモニック（Mnemonic）で記述されている。

【０１８６】まず、この図３８に示されているシンタックスにおいて使用されている関数について説明する。

【０１８７】next_start_code()関数は、ビットストリーム中に記述されているスタートコードを探すための関数である。よって、この図３８に示されたシンタックスにおいて、このnext_start_code()関数の次に、sequence_header()関数とsequence_extension()関数とが順に配置されているので、このビットストリームには、このsequence_header()関数とsequence_extension()関数によって定義されたデータエレメントが記述されている。従って、ビットストリームのデコード時には、このnext_start_code()関数によって、sequence_header()関数とsequence_extension()関数の先頭に記述されているスタートコード（データエレメントの一種）をビットストリーム中から見つけ、それを基準にして、 sequence_header()関数とsequence_extension()関数をさらに見つけ、それらによって定義された各データエレメントをデコードする。

【０１８８】尚、sequence_header()関数は、ＭＰＥＧビットストリームのシーケンス層のヘッダデータを定義するための関数であって、sequence_extension()関数は、ＭＰＥＧビットストリームのシーケンス層の拡張データを定義するための関数である。

【０１８９】sequence_extension()関数の次に配置されている do{ }while構文は、while文によって定義されている条件が真である間、do文の{ }内の関数に基いて記述されたデータエレメントをデータストリーム中から抽出するための構文である。すなわち、 do{ }while構文によって、while文によって定義されている条件が真である間、ビットストリーム中から、do文内の関数に基いて記述されたデータエレメントを抽出するデコード処理が行われる。

【０１９０】このwhile文に使用されているnextbits()関数は、ビットストリーム中に現れるビット又はビット列と、次にデコードされるデータエレメントとを比較するための関数である。この図３８のシンタックスの例では、nextbits()関数は、ビットストリーム中のビット列とビデオシーケンスの終わりを示すsequence_end_codeとを比較し、ビットストリーム中のビット列とsequence_end_codeとが一致しないときに、このwhile文の条件が真となる。従って、sequence_extension()関数の次に配置されている do{ }while構文は、ビットストリーム中に、ビデオシーケンスの終わりを示すsequence_end_codeが現れない間、do文中の関数によって定義されたデータエレメントがビットストリーム中に記述されていることを示している。

【０１９１】ビットストリーム中には、sequence_extension()関数によって定義された各データエレメントの次には、extension_and_user_data(0)関数によって定義されたデータエレメントが記述されている。このextension_and_user_data(0)関数は、ＭＰＥＧビットストリームのシーケンス層の拡張データとユーザデータを定義するための関数である。

【０１９２】このextension_and_user_data(0)関数の次に配置されている do{ }while構文は、while文によって定義されている条件が真である間、do文の{ }内の関数に基いて記述されたデータエレメントを、ビットストリーム中から抽出するための関数である。このwhile文において使用されているnextbits()関数は、ビットストリーム中に現れるビット又はビット列と、picture_start_code又はgroup_start_codeとの一致を判断するための関数であって、ビットストリーム中に現れるビット又はビット列と、picture_start_code又はgroup_start_codeと

が一致する場合には、while文によって定義された条件が真となる。よって、このdo{ }while構文は、ビットストリーム中において、picture_start_code又はgroup_start_codeが現れた場合には、そのスタートコードの次に、do文中の関数によって定義されたデータエレメントのコードが記述されているので、このpicture_start_code又はgroup_start_codeによって示されるスタートコードを探し出すことによって、ビットストリーム中からdo文中に定義されたデータエレメントを抽出することができる。

【０１９３】このdo文の最初に記述されているif文は、ビットストリーム中にgroup_start_codeが現れた場合、という条件を示しいる。このif文による条件が真である場合には、ビットストリーム中には、このgroup_start_codeの次にgroup_of_picture_header(1)関数及びextension_and_user_data(1)関数によって定義されているデータエレメントが順に記述されている。

【０１９４】このgroup_of_picture_header(1)関数は、ＭＰＥＧビットストリームのＧＯＰ層のヘッダデータを定義するための関数であって、 extension_and_user_data(1)関数は、ＭＰＥＧビットストリームのＧＯＰ層の拡張データ（extension_data）及びユーザデータ（user_data）を定義するための関数である。

【０１９５】さらに、このビットストリーム中には、group_of_picture_header(1)関数及びextension_and_user_data(1)関数によって定義されているデータエレメントの次に、picture_header()関数とpicture_coding_extension()関数によって定義されたデータエレメントが記述されている。もちろん、先に説明したif文の条件が真とならない場合には、 group_of_picture_header(1)関数及びextension_and_user_data(1)関数によって定義されているデータエレメントは記述されていないので、 extension_and_user_data(0)関数によって定義されているデータエレメントの次に、 picture_header()関数とpicture_coding_extension()関数によって定義されたデータエレメントが記述されている。

【０１９６】このpicture_header()関数は、 ＭＰＥＧビットストリームのピクチャ層のヘッダデータを定義するための関数であって、 picture_coding_extension()関数は、ＭＰＥＧビットストリームのピクチャ層の第１の拡張データを定義するための関数である。

【０１９７】次のwhile文は、このwhile文によって定義されている条件が真である間、次のif文の条件判断を行うための関数である。このwhile文において使用されているnextbits()関数は、ビットストリーム中に現れるビット列と、extension_start_code又はuser_data_start_codeとの一致を判断するための関数であって、ビットストリーム中に現れるビット列と、 extension_start_code又はuser_data_start_codeとが一致する場合には、このwhile文によって定義された条件が真となる。

【０１９８】第１のif文は、ビットストリーム中に現れるビット列とextension_start_codeとの一致を判断するための関数である。ビットストリーム中に現れるビット列と３２ビットのextension_ start_codeとが一致する場合には、ビットストリーム中において、extension_start_codeの次にextension_data(2)関数によって定義されるデータエレメントが記述されていることを示している。

【０１９９】第２のif文は、ビットストリーム中に現れるビット列とuser_data_start_codeとの一致を判断するための構文であって、ビットストリーム中に現れるビット列と３２ビットのuser_data_start_codeとが一致する場合には、第３のif文の条件判断が行われる。このuser_data_start_codeは、ＭＰＥＧビットストリームのピクチャ層のユーザデータエリアの開始を示すためのスタートコードである。

【０２００】第３のif文は、ビットストリーム中に現れるビット列とHistory_Data_IDとの一致を判断するための構文である。ビットストリーム中に現れるビット列とこの８ビットのHistory_Data_IDとが一致する場合には、このＭＰＥＧビットストリームのピクチャ層のユーザデータエリアにおいて、この８ビットのHistory_Data_IDによって示されるコードの次に、converted_history_stream()関数によって定義されるデータエレメントが記述されていることを示している。

【０２０１】converted_history_stream()関数は、ＭＰＥＧ符号化時に使用したあらゆる符号化パラメータを伝送するための履歴情報及び履歴データを記述するための関数である。このconverted_history_stream()関数によって定義されているデータエレメントの詳細は後述する。また、このHistory_Data_IDは、ＭＰＥＧビットストリームのピクチャ層のユーザデータエリアに記述されたこの履歴情報及び履歴データが記述されている先頭を示すためのスタートコードである。

【０２０２】else文は、第３のif文において、条件が非真であることを示すための構文である。従って、このＭＰＥＧビットストリームのピクチャ層のユーザデータエリアにおいて、converted_history_stream()関数によって定義されたデータエレメントが記述されていない場合には、user_data()関数によって定義されたデータエレメントが記述されている。

【０２０３】picture_data()関数は、ＭＰＥＧビットストリームのピクチャ層のユーザデータの次に、スライス層及びマクロブロック層に関するデータエレメントを記述するための関数である。通常は、このpicture_data()関数によって示されるデータエレメントは、ビットストリームのピクチャ層のユーザデータエリアに記述されたconverted_history_stream()関数によって定義されるデータエレメント又はuser_data()関数によって定義されたデータエレメントの次に記述されているが、ピクチャ

層のデータエレメントを示すビットストリーム中に、extension_start_code又はuser_data_start_code が存在しない場合には、このpicture_data()関数によって示されるデータエレメントは、 picture_coding_extension()関数によって定義されるデータエレメントの次に記述されるている。

【０２０４】このpicture_data()関数によって示されるデータエレメントの次には、sequence_header()関数とsequence_extension()関数とによって定義されたデータエレメントが順に配置されている。このsequence_header()関数とsequence_extension()関数によって記述されたデータエレメントは、ビデオストリームのシーケンスの先頭に記述されたsequence_header()関数とsequence_extension()関数によって記述されたデータエレメントと全く同じである。このように同じデータをストリーム中に記述する理由は、ビットストリーム受信装置側でデータストリームの途中（例えばピクチャ層に対応するビットストリーム部分）から受信が開始された場合に、シーケンス層のデータを受信できなくなり、ストリームをデコード出来なくなることを防止するためである。

【０２０５】この最後のsequence_header()関数とsequence_extension()関数とによって定義されたデータエレメントの次、つまり、データストリームの最後には、シーケンスの終わりを示す３２ビットのsequence_end_codeが記述されている。

【０２０６】以上のシンタックスの基本的な構成の概略を示すと、図３９に示すようになる。

【０２０７】次に、converted_history_stream()関数によって定義されたヒストリーストリームに関して説明する。

【０２０８】このconverted_history_stream()は、ＭＧＥＧのピクチャ層のユーザデータエリアに履歴情報を示すヒストリーストリームを挿入するための関数である。尚、「converted」の意味は、スタートエミュレーションを防止するために、ユーザエリアに挿入すべき履歴データから構成される履歴ストリームの少なくとも２２ビット毎にマーカービット（１ビット）を挿入する変換処理を行ったストリームであることを意味している。

【０２０９】このconverted_history_stream()は、以下に説明する固定長の履歴ストリーム（図４０乃至図４６）又は可変長の履歴ストリーム（図４７）のいずれかの形式で記述される。エンコーダ側において固定長の履歴ストリームを選択した場合には、デコーダ側において履歴ストリームから各データエレメントをデコードするための回路及びソフトウエアが簡単になるというメリットがある。一方、エンコーダ側において可変長の履歴ストリームを選択した場合には、エンコーダにおいてピクチャ層のユーザエリアに記述される履歴情報（データエレメント）を必要に応じて任意に選択することができるので、履歴ストリームのデータ量を少なくすることができ、その結果、符号化されたビットストリーム全体のデータレートを低減することができる。

【０２１０】本発明において説明する「履歴情報」「履歴データ」「履歴パラメータ」とは、過去の符号化処理において使用した符号化パラメータ（又はデータエレメント）のことであって、現在の（最終段の）符号化処理において使用した符号化パラメータのことではない。例えば、第１世代の符号化処理において、あるピクチャをＩピクチャで符号化して伝送し、次なる第２世代の符号化処理において、このピクチャを今度はＰピクチャとして符号化して伝送し、さらに、第３世代の符号化処理において、このピクチャをＢピクチャで符号化して伝送する例をあげて説明する。第３世代の符号化処理において使用した符号化パラメータが、第３世代の符号化処理において生成された符号化ビットストリームのシーケンス層、ＧＯＰ層、ピクチャ層、スライス層及びマクロブロック層の所定位置に記述されている。一方、過去の符号化処理である第１世代及び第２世代の符号化処理において使用した符号化パラメータは、第３世代の符号化処理において使用した符号化パラメータが記述されるシーケンス層やＧＯＰ層に記述されるのでは無く、既に説明したシンタックスに従って、符号化パラメータの履歴情報として、ピクチャ層のユーザデータエリアに記述される。

【０２１１】まず、固定長の履歴ストリームシンタックスについて図４０乃至図４６を参照して説明する。

【０２１２】最終段（例えば第３世代）の符号化処理において生成されたビットストリームのピクチャ層のユーザエリアには、まず最初に、過去（例えば第１世代及び第２世代）の符号化処理において使用されていたシーケンス層のシーケンスヘッダに関する符号化パラメータが、履歴ストリームとして挿入される。尚、過去の符号化処理において生成されたビットストリームのシーケンス層のシーケンスヘッダ等の履歴情報は、最終段の符号化処理において生成されたビットストリームのシーケンス層のシーケンスヘッダに挿入されることは無いという点に注意すべきである。

【０２１３】過去の符号化処理で使用したシーケンスヘッダに関するデータエレメントは、sequence_header_code、sequence_header_present_flag、horizontal_size_value、vertical_size_value、aspect_ratio_information、frame_rate_code、bit_rate_value、marker_bit、VBV_buffer_size_value、constrained_parameter_flag、load_intra_quantizer_matrix、intra_quantizer_matrix、load_non_intra_quantizer_matrix、及びnon_intra_quantizer_matrix等から構成される。

【０２１４】sequence_header_codeは、シーケンス層のスタート同期コードを表すデータである。sequence_header_present_flagは、sequence_header内のデータが有効か無効かを示すデータである。 horizontal_size_val

ueは、画像の水平方向の画素数の下位12ビットから成るデータである。vertical_size_valueは、画像の縦のライン数の下位12ビットからなるデータである。aspect_ratio_informationは、画素のアスペクト比（縦横比）または表示画面アスペクト比を表すデータである。frame_rate_codeは、画像の表示周期を表すデータである。

【０２１５】bit_rate_valueは、発生ビット量に対する制限のためのビット・レートの下位18ビット(400bsp単位で切り上げる)データである。marker_bitは、スタートコードエミュレーションを防止するために挿入されるビットデータである。VBV_buffer_size_valueは、発生符号量制御用の仮想バッファ（ビデオバッファベリファイヤー）の大きさを決める値の下位10ビットデータである。constrained_parameter_flagは、各パラメータが制限以内であることを示すデータである。load_intra_quantizer_matrixは、イントラMB用量子化マトリックス・データの存在を示すデータである。intra_quantizer_matrixは、イントラＭＢ用量子化マトリックスの値を示すデータである。load_non_intra_quantizer_matrixは、非イントラＭＢ用量子化マトリックス・データの存在を示すデータである。non_intra_quantizer_matrixは、非イントラＭＢ用量子化マトリックスの値を表すデータである。

【０２１６】次に、最終段の符号化処理において生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたシーケンス層のシーケンスエクステンションを表わすデータエレメントが、履歴ストリームとして記述される。

【０２１７】この過去の符号化処理で使用したシーケンスエクステンションを表わすデータエレメントは、 extension_start_code、extension_start_code_identifier、sequence_extension_present_flag、profile_and_level_indication、progressive_sequence、chroma_format、horizontal_size_extension、vertical_size_extension、bit_rate_extension、vbv_buffer_size_extension、low_delay、frame_rate_extension_n 、及び frame_rate_extension_d等のデータエレメントである。

【０２１８】extension_start_codeは、エクステンションデータのスタート同期コードを表すデータである。extension_start_code_identifierは、どの拡張データが送られるかを示すデータである。sequence_extension_present_flagは、シーケンスエクステンション内のデータが有効であるか無効であるかを示すデータである。profile_and_level_indicationは、ビデオデータのプロファイルとレベルを指定するためのデータである。progressive_sequenceは、ビデオデータが順次走査であることを示すデータである。chroma_formatは、ビデオデータの色差フォーマットを指定するためのデータである。

【０２１９】horizontal_size_extensionは、シーケンスヘッダのhoriztal_size_valueに加える上位２ビットのデータである。vertical_size_extensionは、シーケンスヘッダのvertical_size_valueに加える上位２ビットのデータである。bit_rate_extensionは、シーケンスヘッダのbit_rate_valueに加える上位１２ビットのデータである。vbv_buffer_size_extensionは、シーケンスヘッダのvbv_buffer_size_valueに加える上位８ビットのデータである。low_delayは、Ｂピクチャを含まないことを示すデータである。frame_rate_extension_nは、シーケンスヘッダのframe_rate_codeと組み合わせてフレームレートを得るためのデータである。frame_rate_extension_dは、シーケンスヘッダのframe_rate_codeと組み合わせてフレームレートを得るためのデータである。

【０２２０】続いて、ビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたシーケンス層のシーケンスディスプレイエクステンションを表わすデータエレメントが、履歴ストリームとして記述される。

【０２２１】このシーケンスディスプレイエクステンションとして記述されているデータエレメントは、extension_start_code、extension_start_code_identifier、sequence_display_extension_present_flag、video_format、color_description、color_primaries、transfer_characteristics、matrix_coeffients、display_horizontal_size、及びdisplay_vertical_sizeから構成される。

【０２２２】extension_start_codeは、エクステンションデータのスタート同期コードを表すデータである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。sequence_display_extension_present_flagは、シーケンスディスプレイエクステンション内のデータエレメントが有効か無効かを示すデータである。video_formatは、原信号の映像フォーマットを表すデータである。color_descriptionは、色空間の詳細データがあることを示すデータである。color_primariesは、原信号の色特性の詳細を示すデータである。transfer_characteristicsは、光電変換がどのように行われたのかの詳細を示すデータである。matrix_coeffientsは、原信号が光の三原色からどのように変換されたかの詳細を示すデータである。display_horizontal_sizeは、意図するディスプレイの活性領域（水平サイズ）を表すデータである。display_vertical_sizeは、意図するディスプレイの活性領域（垂直サイズ）を表すデータである。

【０２２３】続いて、最終段の符号化処理において生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において生成されたマクロブロックの位相情報を示すマクロブロックアサイメントデータ（macroblock_assignment_in_user_data）が、履歴ストリームとして記述される。

【０２２４】このマクロブロックの位相情報を示すmacroblock_assignment_in_user_dataは、macroblock_assignment_present_flag、v_phase、h_phase等のデータエレメントから構成される。

【０２２５】このmacroblock_assignment_present_flagは、macroblock_assignment_in_user_data内のデータエレメントが有効か無効かを示すデータである。 v_phaseは、画像データからマクロブロックを切り出す際の垂直方向の位相情報を示すデータである。 h_phaseは、画像データからマクロブロックを切り出す際の水平方向の位相情報を示すデータである。

【０２２６】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたＧＯＰ層のＧＯＰヘッダを表わすデータエレメントが、履歴ストリームとして記述されている。

【０２２７】このＧＯＰヘッダを表わすデータエレメントは、group_start_code、group_of_picture_header_present_flag、time_code、closed_gop、及びbroken_linkから構成される。

【０２２８】group_start_codeは、ＧＯＰ層の開始同期コードを示すデータである。 group_of_picture_header_present_flagは、 group_of_picture_header内のデータエレメントが有効であるか無効であるかを示すデータである。 time_codeは、ＧＯＰの先頭ピクチャのシーケンスの先頭からの時間を示すタイムコードである。closed_gopは、ＧＯＰ内の画像が他のＧＯＰから独立再生可能なことを示すフラグデータである。broken_linkは、編集などのためにＧＯＰ内の先頭のＢピクチャが正確に再生できないことを示すフラグデータである。

【０２２９】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたピクチャ層のピクチャヘッダを表わすデータエレメントが、履歴ストリームとして記述されている。

【０２３０】このピクチャヘッダに関するデータエレメントは、picture_start_code、temporal_reference、picture_coding_type、vbv_delay、full_pel_forward_vector、forward_f_code、full_pel_backward_vector、及び backward_f_codeから構成される。

【０２３１】具体的には、picture_start_codeは、ピクチャ層の開始同期コードを表すデータである。temporal_referenceは、ピクチャの表示順を示す番号でＧＯＰの先頭でリセットされるデータである。picture_coding_typeは、ピクチャタイプを示すデータである。vbv_delayは、ランダムアクセス時の仮想バッファの初期状態を示すデータである。full_pel_forward_vectorは、順方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。forward_f_codeは、順方向動きベクトル探索範囲を表すデータである。full_pel_backward_vectorは、逆方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。backward_f_codeは、逆方向動きベクトル探索範囲を表すデータである。

【０２３２】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたピクチャ層のピクチャコーディングエクステンションが、履歴ストリームとして記述されている。

【０２３３】このピクチャコーディングエクステンションに関するデータエレメントは、extension_start_code、extension_start_code_identifier、f_code[0][0]、f_code[0][1]、f_code[1][0]、f_code[1][1]、intra_dc_precision、picture_structure、top_field_first、frame_predictive_frame_dct、concealment_motion_vectors、q_scale_type、intra_vlc_format、alternate_scan、repeat_firt_field、chroma_420_type、progressive_frame、composite_display_flag、v_axis、field_sequence、sub_carrier、burst_amplitude、及びsub_carrier_phaseから構成される。

【０２３４】extension_start_codeは、ピクチャ層のエクステンションデータのスタートを示す開始コードである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。 f_code[0][0]は、フォワード方向の水平動きベクトル探索範囲を表すデータである。f_code[0][1]は、フォワード方向の垂直動きベクトル探索範囲を表すデータである。f_code[1][0]は、バックワード方向の水平動きベクトル探索範囲を表すデータである。f_code[1][1]は、バックワード方向の垂直動きベクトル探索範囲を表すデータである。

【０２３５】intra_dc_precisionは、DC係数の精度を表すデータである。picture_structureは、フレームストラクチャかフィールドストラクチャかを示すデータである。フィールドストラクチャの場合は、上位フィールドか下位フィールドかもあわせて示すデータである。top_field_firstは、フレームストラクチャの場合、最初のフィールドが上位か下位かを示すデータである。frame_predictive_frame_dctは、フレーム・ストラクチャの場合、フレーム・モードＤＣＴの予測がフレーム・モードだけであることを示すデータである。concealment_motion_vectorsは、イントラマクロブロックに伝送エラーを隠蔽するための動きベクトルがついていることを示すデータである。

【０２３６】q_scale_typeは、線形量子化スケールを利用するか、非線形量子化スケールを利用するかを示すデータである。intra_vlc_formatは、イントラマクロブロックに、別の２次元ＶＬＣを使うかどうかを示すデータである。alternate_scanは、ジグザグスキャンを使うか、オルタネート・スキャンを使うかの選択を表すデータである。repeat_firt_fieldは、２：３プルダウンの際に使われるデータである。chroma_420_typeは、信号

フォーマットが４：２：０の場合、次のprogressive_frame と同じ値、そうでない場合は０を表すデータである。progressive_frameは、このピクチャが、順次走査できているかどうかを示すデータである。composite_display_flagは、ソース信号がコンポジット信号であったかどうかを示すデータである。

【０２３７】v_axisは、ソース信号が、ＰＡＬの場合に使われるデータである。field_sequenceは、ソース信号が、ＰＡＬの場合に使われるデータである。sub_carrierは、ソース信号が、ＰＡＬの場合に使われるデータである。burst_amplitudeは、ソース信号が、ＰＡＬの場合に使われるデータである。sub_carrier_phaseは、ソース信号が、ＰＡＬの場合に使われるデータである。

【０２３８】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用された量子化マトリックスエクステンションが、履歴ストリームとして記述されている。

【０２３９】この量子化マトリックスエクステンションに関するデータエレメントは、extension_start_code、extension_start_code_identifier、quant_matrix_extension_present_flag、load_intra_quantizer_matrix、intra_quantizer_matrix[64]、load_non_intra_quantizer_matrix、non_intra_quantizer_matrix[64]、load_chroma_intra_quantizer_matrix、chroma_intra_quantizer_matrix[64]、load_chroma_non_intra_quantizer_matrix、及びchroma_non_intra_quantizer_matrix[64] から構成される。

【０２４０】extension_start_codeは、この量子化マトリックスエクステンションのスタートを示す開始コードである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。 quant_matrix_extension_present_flagは、この量子化マトリックスエクステンション内のデータエレメントが有効か無効かを示すためのデータである。load_intra_quantizer_matrixは、イントラマクロブロック用の量子化マトリックスデータの存在を示すデータである。intra_quantizer_matrixは、イントラマクロブロック用の量子化マトリックスの値を示すデータである。

【０２４１】load_non_intra_quantizer_matrixは、非イントラマクロブロック用の量子化マトリックスデータの存在を示すデータである。non_intra_quantizer_matrixは、非イントラマクロブロック用の量子化マトリックスの値を表すデータである。load_chroma_intra_quantizer_matrixは、色差イントラマクロブロック用の量子化マトリックス・データの存在を示すデータである。chroma_intra_quantizer_matrixは、色差イントラマクロブロック用の量子化マトリックスの値を示すデータである。load_chroma_non_intra_quantizer_matrixは、色差非イントラマクロブロック用の量子化マトリックス・データの存在を示すデータである。chroma_non_intra_quantizer_matrixは、色差非イントラマクロブロック用の量子化マトリックスの値を示すデータである。

【０２４２】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたコピーライトエクステンションが、履歴ストリームとして記述されている。

【０２４３】このコピーライトエクステンションに関するデータエレメントは、extension_start_code、extension_start_code_itentifier、copyright_extension_present_flag、copyright_flag、copyright_identifier、original_or_copy、copyright_number_1、copyright_number_2、及び copyright_number_3から構成される。

【０２４４】extension_start_codeは、コピーライトエクステンションのスタート示す開始コードである。extension_start_code_itentifierのどのエクステンションデータが送られるかを示すコードである。 copyright_extension_present_flagは、このコピーライトエクステンション内のデータエレメントが有効か無効かを示すためのデータである。copyright_flagは、次のコピーライトエクステンション又はシーケンスエンドまで、符号化されたビデオデータに対してコピー権が与えられているか否かを示す。

【０２４５】copyright_identifierは、ＩＳＯ／ＩＥＣＪＴＣ／ＳＣ２９によって指定されたコピー権の登録機関を識別するためのデータである。original_or_copyは、ビットストリーム中のデータが、オリジナルデータであるかコピーデータであるかを示すデータである。copyright_number_1は、コピーライトナンバーのビット４４から６３を表わすデータである。copyright_number_2は、コピーライトナンバーのビット２２から４３を表わすデータである。copyright_number_3は、コピーライトナンバーのビット０から２１を表わすデータである。

【０２４６】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたピクチャディスプレイエクステンション（ picture_display_extension ）が、履歴ストリームとして記述されている。

【０２４７】このピクチャディスプレイエクステンションを表わすデータエレメントは、extension_start_code、extension_start_code_identifier、picture_display_extension_present_flag、frame_center_horizontal_offset_1、frame_center_vertical_offset_1、frame_center_horizontal_offset_2、frame_center_vertical_offset_2、frame_center_horizontal_offset_3、及びframe_center_vertical_offset_3から構成される。

【０２４８】extension_start_codeは、ピクチャディスプレイエクステンションのスタートを示すための開始コードである。extension_start_code_identifierは、ど

の拡張データが送られるかを示すコードである。picture_display_extension_present_flagは、ピクチャディスプレイエクステンション内のデータエレメントが有効か無効かを示すデータである。frame_center_horizontal_offsetは、表示エリアの水平方向のオフセットを示すデータであって、３つのオフセット値まで定義することができる。frame_center_vertical_offsetは、表示エリアを垂直方向のオフセットを示すデータであって、３つのオフセット値まで定義することができる。

【０２４９】最終段の符号化処理において生成されたビットストリームのピクチャ層のユーザエリアには、既に説明したピクチャディスプレイエクステンションを表わす履歴情報の次に、過去の符号化処理において使用されたユーザデータが、履歴ストリームとして記述されている。

【０２５０】このユーザデータの次には、過去の符号化処理において使用されたマクロブロック層に関する情報が、履歴ストリームとして記述されている。

【０２５１】このマクロブロック層に関する情報は、macroblock_address_h、macroblock_address_v、slice_header_present_flag、skipped_macroblock_flag等のマクロブロックの位置に関するデータエレメントと、macroblock_quant、macroblock_motion_forward、macroblock_motion_backward、mocroblock_pattern、macroblock_intra、spatial_temporal_weight_code_flag、frame_motion_type、及びdct_type等のマクロブロックモードに関するデータエレメントと、quantiser_scale_code等の量子化ステップ制御に関するデータエレメントと、PMV[0][0][0]、PMV[0][0][1]、motion_vertical_field_select[0][0]、PMV[0][1][0]、PMV[0][1][1]、motion_vertical_field_select[0][1]、PMV[1][0][0]、PMV[1][0][1]、motion_vertical_field_select[1][0]、PMV[1][1][0]、PMV[1][1][1]、motion_vertical_field_select[1][1]等の動き補償に関するデータエレメントと、coded_block_pattern等のマクロブロックパターンに関するデータエレメントと、num_mv_bits、num_coef_bits、及びnum_other_bits等の発生符号量に関するデータエレメントから構成されている。

【０２５２】以下にマクロブロック層に関するデータエレメントについて詳細に説明する。

【０２５３】macroblock_address_hは、現在のマクロブロックの水平方向の絶対位置を定義するためのデータである。macroblock_address_vは、現在のマクロブロックの垂直方向の絶対位置を定義するためのデータである。slice_header_present_flagは、このマクロブロックがスライス層の先頭であり、スライスヘッダを伴なうか否かを示すデータである。skipped_macroblock_flagは、復号化処理においてこのマクロブロックをスキップするか否かを示すデータでる。

【０２５４】macroblock_quantは、後述する図６５乃至図６７に示されたマクロブロックタイプ（ macroblock_type ）から導かれるデータであって、quantiser_scale_codeがビットストリーム中に現れるか否かを示すデータである。macroblock_motion_forwardは、図６５乃至図６７に示されたマクロブロックタイプから導かれるデータであって、復号化処理で使用されるデータである。macroblock_motion_backwardは、図６５乃至図６７に示されたマクロブロックタイプから導かれるデータであって、復号化処理で使用されるデータである。mocroblock_patternは、図６５乃至図６７に示されたマクロブロックタイプから導かれるデータであって、coded_block_patternがビットストリーム中に現れるか否かを示すデータである。

【０２５５】macroblock_intraは、図６５乃至図６７に示されたマクロブロックタイプから導かれるデータであって、復号化処理で使用されるデータである。spatial_temporal_weight_code_flagは、図６５乃至図６７に示されたマクロブロックタイプから導かれるデータであって、時間スケーラビリティで下位レイヤ画像のアップサンプリング方法を示すspatial_temporal_weight_codeは、ビットストリーム中に存在するか否かを示すデータである。

【０２５６】frame_motion_typeは、フレームのマクロブロックの予測タイプを示す２ビットのコードである。予測ベクトルが２個でフィールドベースの予測タイプであれば「００」であって、予測ベクトルが１個でフィールドベースの予測タイプであれば「０１」であって、予測ベクトルが１個でフレームベースの予測タイプであれば「１０」であって、予測ベクトルが１個でディアルプライムの予測タイプであれば「１１」である。field_motion_typeは、フィールドのマクロブロックの動き予測を示す２ビットのコードである。予測ベクトルが１個でフィールドベースの予測タイプであれば「０１」であって、予測ベクトルが２個で１８×８マクロブロックベースの予測タイプであれば「１０」であって、予測ベクトルが１個でディアルプライムの予測タイプであれば「１１」である。dct_typeは、ＤＣＴがフレームＤＣＴモードか、フィールドＤＣＴモードかを示すデータである。quantiser_scale_codeはマクロブロックの量子化ステップサイズを示すデータである。

【０２５７】次に動きベクトルに関するデータエレメントについて説明する。動きベクトルは、復号時に必要な動きベクトルを減少させるために、先に符号化されたベクトルに関し差分として符号化される。動きベクトルの復号を行うために復号器は、４個の動きベクトル予測値（それぞれ水平及び垂直成分を伴なう）を維持しなければいけない。この予測動きベクトルをPMV[r][s][v]と表わすことにしている。[r]は、マクロブロックにおける動きベクトルが第１のベクトルであるのか、第２のベクトルであるのかを示すフラグであって、マクロブロック

におけるベクトルが第１のベクトルである場合には「０」となって、マクロブロックにおけるベクトルが第２のベクトルである場合には「１」となる。[s]は、マクロブロックにおける動きベクトルの方向が、前方向であるのか後方向であるのかを示すフラグであって、前方向動きベクトルの場合には「０」となって、後方向動きベクトルの場合には「１」となる。[v]は、マクロブロックにおけるベクトルの成分が、水平方向であるのか垂直方向であるのかを示すフラグであって、水平方向成分の場合には「０」となって、垂直方向成分の場合には「１」となる。

【０２５８】従って、PMV[0][0][0]は、第１のベクトルの前方向の動きベクトルの水平方向成分のデータを表わし、PMV[0][0][1]は、第１のベクトルの前方向の動きベクトルの垂直方向成分のデータを表わし、PMV[0][1][0]は、第１のベクトルの後方向の動きベクトルの水平方向成分のデータを表わし、PMV[0][1][1]は、第１のベクトルの後方向の動きベクトルの垂直方向成分のデータを表わし、 PMV[1][0][0]は、第２のベクトルの前方向の動きベクトルの水平方向成分のデータを表わし、PMV[1][0][1]は、第２のベクトルの前方向の動きベクトルの垂直方向成分のデータを表わし、 PMV[1][1][0]は、第２のベクトルの後方向の動きベクトルの水平方向成分のデータを表わし、PMV[1][1][1] は、第２のベクトルの後方向の動きベクトルの垂直方向成分のデータを表わしている。

【０２５９】motion_vertical_field_select[r][s]は、予測の形式にいずれの参照フィールドを使用するのかを示すデータである。このmotion_vertical_field_select[r][s]が「０」の場合には、トップ参照フィールドを使用し、「１」の場合には、ボトム参照フィールドを使用することを示している。

【０２６０】よって、motion_vertical_field_select[0][0]は、第１のベクトルの前方向の動きベクトルを生成する際の参照フィールドを示し、motion_vertical_field_select[0][1]は、第１のベクトルの後方向の動きベクトルを生成する際の参照フィールドを示し、motion_vertical_field_select[1][0]は、第２のベクトルの前方向の動きベクトルを生成する際の参照フィールドを示し、motion_vertical_field_select[1][1]は、第２ベクトルの後方向の動きベクトルを生成する際の参照フィールドを示している。

【０２６１】coded_block_patternは、ＤＣＴ係数を格納する複数のＤＣＴブロックのうち、どのＤＣＴブロックに、有意係数（非０係数）があるかを示す可変長のデータである。num_mv_bitsは、マクロブロック中の動きベクトルの符号量を示すデータである。num_coef_bitsは、マクロブロック中のＤＣＴ係数の符号量を示すデータである。num_other_bitsは、マクロブロックの符号量で、動きベクトル及びＤＣＴ係数以外の符号量を示すデータである。

【０２６２】次に、可変長の履歴ストリームから各データエレメントをデコードするためのシンタックスについて、図４７乃至図６４を参照して説明する。

【０２６３】この可変長の履歴ストリームは、next_start_code()関数、sequence_header()関数、sequence_extension()関数、extension_and_user_data(0)関数、group_of_picture_header()関数、extension_and_user_data(1)関数、picture_header()関数、picture_coding_extension()関数、extension_and_user_data(2)関数、及びpicture_data()関数によって定義されたデータエレメントによって構成される。

【０２６４】next_start_code()関数は、ビットストリーム中に存在するスタートコードを探すための関数であるので、履歴ストリームの最も先頭には、図４８に示すような、過去の符号化処理において使用されたデータエレメントであってsequence_header()関数によって定義されたデータエレメントが記述されている。

【０２６５】sequence_header()関数によって定義されたデータエレメントは、sequence_header_code、sequence_header_present_flag、horizontal_size_value、vertical_size_value、aspect_ratio_information、frame_rate_code、bit_rate_value、marker_bit、VBV_buffer_size_value、constrained_parameter_flag、load_intra_quantizer_matrix、intra_quantizer_matrix、load_non_intra_quantizer_matrix、及びnon_intra_quantizer_matrix等である。

【０２６６】sequence_header_codeは、シーケンス層のスタート同期コードを表すデータである。sequence_header_present_flagは、sequence_header内のデータが有効か無効かを示すデータである。 horizontal_size_valueは、画像の水平方向の画素数の下位12ビットから成るデータである。vertical_size_valueは、画像の縦のライン数の下位12ビットからなるデータである。aspect_ratio_informationは、画素のアスペクト比（縦横比）または表示画面アスペクト比を表すデータである。frame_rate_codeは、画像の表示周期を表すデータである。bit_rate_valueは、発生ビット量に対する制限のためのビット・レートの下位18ビット(400bsp単位で切り上げる)データである。

【０２６７】marker_bitは、スタートコードエミュレーションを防止するために挿入されるビットデータである。VBV_buffer_size_valueは、発生符号量制御用の仮想バッファ（ビデオバッファベリファイヤー）の大きさを決める値の下位10ビットデータである。constrained_parameter_flagは、各パラメータが制限以内であることを示すデータである。load_intra_quantizer_matrixは、イントラMB用量子化マトリックス・データの存在を示すデータである。intra_quantizer_matrixは、イントラＭＢ用量子化マトリックスの値を示すデータである。

load_non_intra_quantizer_matrixは、非イントラＭＢ用量子化マトリックス・データの存在を示すデータである。non_intra_quantizer_matrixは、非イントラＭＢ用量子化マトリックスの値を表すデータである。

【０２６８】sequence_header()関数によって定義されたデータエレメントの次には、図４９で示すような、sequence_extension()関数によって定義されたデータエレメントが、履歴ストリームとして記述されている。

【０２６９】sequence_extension()関数によって定義されたデータエレメントとは、extension_start_code、extension_start_code_identifier、sequence_extension_present_flag、profile_and_level_indication、progressive_sequence、chroma_format、horizontal_size_extension、vertical_size_extension、bit_rate_extension、vbv_buffer_size_extension、low_delay、frame_rate_extension_n 、及びframe_rate_extension_d等のデータエレメントである。

【０２７０】extension_start_codeは、エクステンションデータのスタート同期コードを表すデータである。extension_start_code_identifierは、どの拡張データが送られるかを示すデータである。sequence_extension_present_flagは、シーケンスエクステンション内のデータが有効であるか無効であるかを示すスデータである。profile_and_level_indicationは、ビデオデータのプロファイルとレベルを指定するためのデータである。progressive_sequenceは、ビデオデータが順次走査であることを示すデータである。chroma_formatは、ビデオデータの色差フォーマットを指定するためのデータである。horizontal_size_extensionは、シーケンスヘッダのhoriztal_size_valueに加える上位２ビットのデータである。vertical_size_extensionは、シーケンスヘッダのvertical_size_value加える上位２ビットのデータである。bit_rate_extensionは、シーケンスヘッダのbit_rate_valueに加える上位１２ビットのデータである。vbv_buffer_size_extensionは、シーケンスヘッダのvbv_buffer_size_valueに加える上位８ビットのデータである。

【０２７１】low_delayは、Ｂピクチャを含まないことを示すデータである。frame_rate_extension_nは、シーケンスヘッダのframe_rate_codeと組み合わせてフレームレートを得るためのデータである。frame_rate_extension_dは、シーケンスヘッダのframe_rate_codeと組み合わせてフレームレートを得るためのデータである。

【０２７２】sequence_extension()関数によって定義されたデータエレメントの次には、図５０に示すようなextension_and_user_data(0)関数によって定義されたデータエレメントが、履歴ストリームとして記述されている。 extension_and_user_data(i)関数は、「i」が2以外のときは、extension_data()関数によって定義されるデータエレメントは記述せずに、user_data()関数によって定義されるデータエレメントのみを履歴ストリームとして記述する。よって、 extension_and_user_data(0)関数は、 user_data()関数によって定義されるデータエレメントのみを履歴ストリームとして記述する。

【０２７３】user_data()関数は、図５１に示されたようなシンタックスに基いて、ユーザデータを履歴ストリームとして記述する。

【０２７４】extension_and_user_data(0)関数によって定義されたデータエレメントの次には、図５２に示すようなgroup_of_picture_header()関数によって定義されたデータエレメント、及びextension_and_user_data(1)関数によって定義されるデータエレメントが、履歴ストリームとして記述されている。但し、履歴ストリーム中に、ＧＯＰ層のスタートコードを示すgroup_start_codeが記述されている場合にのみ、 group_of_picture_header()関数によって定義されたデータエレメント、及びextension_and_user_data(1)関数によって定義されるデータエレメントが記述されている。

【０２７５】group_of_picture_header()関数によって定義されたデータエレメントは、group_start_code、group_of_picture_header_present_flag、time_code、closed_gop、及びbroken_linkから構成される。

【０２７６】group_start_codeは、ＧＯＰ層の開始同期コードを示すデータである。 group_of_picture_header_present_flagは、 group_of_picture_header内のデータエレメントが有効であるか無効であるかを示すデータである。 time_codeは、ＧＯＰの先頭ピクチャのシーケンスの先頭からの時間を示すタイムコードである。closed_gopは、ＧＯＰ内の画像が他のＧＯＰから独立再生可能なことを示すフラグデータである。broken_linkは、編集などのためにＧＯＰ内の先頭のＢピクチャが正確に再生できないことを示すフラグデータである。

【０２７７】extension_and_user_data(1)関数は、 extension_and_user_data(0)関数と同じように、user_data()関数によって定義されるデータエレメントのみを履歴ストリームとして記述する。

【０２７８】もし、履歴ストリーム中に、ＧＯＰ層のスタートコードを示すgroup_start_codeが存在しない場合には、これらのgroup_of_picture_header()関数及びextension_and_user_data(1)関数によって定義されるデータエレメントは、履歴ストリーム中には記述されていない。その場合には、 extension_and_user_data(0)関数によって定義されたデータエレメントの次に、picture_headr()関数によって定義されたデータエレメントが履歴ストリームとして記述されている。

【０２７９】picture_headr()関数によって定義されたデータエレメントは、図５３に示すように、picture_start_code、temporal_reference、picture_coding_type、vbv_delay、full_pel_forward_vector、forward_f_code、full_pel_backward_vector、backward_f_code、extra_bit_picture、及びextra_information_pictureであ

る。

【０２８０】具体的には、picture_start_codeは、ピクチャ層の開始同期コードを表すデータである。temporal_referenceは、ピクチャの表示順を示す番号でＧＯＰの先頭でリセットされるデータである。picture_coding_typeは、ピクチャタイプを示すデータである。vbv_delayは、ランダムアクセス時の仮想バッファの初期状態を示すデータである。full_pel_forward_vectorは、順方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。forward_f_codeは、順方向動きベクトル探索範囲を表すデータである。full_pel_backward_vectorは、逆方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。backward_f_codeは、逆方向動きベクトル探索範囲を表すデータである。 extra_bit_pictureは、後続する追加情報の存在を示すフラグである。このextra_bit_pictureが「１」の場合には、次にextra_information_pictureが存在し、extra_bit_pictureが「０」の場合には、これに続くデータが無いことを示している。extra_information_pictureは、規格において予約された情報である。

【０２８１】picture_headr()関数によって定義されたデータエレメントの次には、図５４に示すようなpicture_coding_extension()関数によって定義されたデータエレメントが、履歴ストリームとして記述されている。

【０２８２】このpicture_coding_extension()関数によって定義されたデータエレメントとは、extension_start_code、extension_start_code_identifier、f_code[0][0]、f_code[0][1]、f_code[1][0]、f_code[1][1]、intra_dc_precision、picture_structure、top_field_first、frame_predictive_frame_dct、concealment_motion_vectors、q_scale_type、intra_vlc_format、alternate_scan、repeat_firt_field、chroma_420_type、progressive_frame、composite_display_flag、v_axis、field_sequence、sub_carrier、burst_amplitude、及びsub_carrier_phaseから構成される。

【０２８３】extension_start_codeは、ピクチャ層のエクステンションデータのスタートを示す開始コードである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。 f_code[0][0]は、フォワード方向の水平動きベクトル探索範囲を表すデータである。f_code[0][1]は、フォワード方向の垂直動きベクトル探索範囲を表すデータである。f_code[1][0]は、バックワード方向の水平動きベクトル探索範囲を表すデータである。f_code[1][1]は、バックワード方向の垂直動きベクトル探索範囲を表すデータである。intra_dc_precisionは、DC係数の精度を表すデータである。

【０２８４】picture_structureは、フレームストラクチャかフィールドストラクチャかを示すデータである。フィールドストラクチャの場合は、上位フィールドか下位フィールドかもあわせて示すデータである。top_field_firstは、フレームストラクチャの場合、最初のフィールドが上位か下位かを示すデータである。frame_predictive_frame_dctは、フレーム・ストラクチャの場合、フレーム・モードＤＣＴの予測がフレーム・モードだけであることを示すデータである。concealment_motion_vectorsは、イントラマクロブロックに伝送エラーを隠蔽するための動きベクトルがついていることを示すデータである。q_scale_typeは、線形量子化スケールを利用するか、非線形量子化スケールを利用するかを示すデータである。intra_vlc_formatは、イントラマクロブロックに、別の２次元ＶＬＣを使うかどうかを示すデータである。

【０２８５】alternate_scanは、ジグザグスキャンを使うか、オルタネート・スキャンを使うかの選択を表すデータである。repeat_firt_fieldは、２：３プルダウンの際に使われるデータである。chroma_420_typeは、信号フォーマットが４：２：０の場合、次のprogressive_frame と同じ値、そうでない場合は０を表すデータである。progressive_frameは、このピクチャが、順次走査できているかどうかを示すデータである。composite_display_flagは、ソース信号がコンポジット信号であったかどうかを示すデータである。v_axisは、ソース信号が、ＰＡＬの場合に使われるデータである。field_sequenceは、ソース信号が、ＰＡＬの場合に使われるデータである。sub_carrierは、ソース信号が、ＰＡＬの場合に使われるデータである。burst_amplitudeは、ソース信号が、ＰＡＬの場合に使われるデータである。sub_carrier_phaseは、ソース信号が、ＰＡＬの場合に使われるデータである。

【０２８６】picture_coding_extension()関数によって定義されたデータエレメントの次には、extensions_and_user_data(2)によって定義されたデータエレメントが、履歴ストリームとして記述されている。このextension_and_user_data(2)関数は、図５０に示したように、ビットストリーム中にエクステンションスタートコード（extension_start_code）が存在する場合には、extension_data()関数によって定義されるデータエレメントが記述されている。このデータエレメントの次には、ビットストリーム中にユーザデータスタートコード（user_data_start_code）が存在する場合には、user_data()関数によって定義されるデータエレメントが記述されている。但し、ビットストリーム中にエクステンションスタートコード及びユーザデータスタートコードが存在しない場合には extension_data()関数及びuser_data()関数によって定義されるデータエレメントはビットストリーム中には記述されていない。

【０２８７】extension_data()関数は、図５５に示すように、extension_start_codeを示すデータエレメントと、quant_matrix_extension()関数、copyright_extens

ion()関数、及びpicture_display_extension()関数によって定義されるデータエレメンエトとを、ビットストリーム中に履歴ストリームとして記述するための関数である。

【０２８８】quant_matrix_extension()関数によって定義されるデータエレメントは、図５６に示すように、extension_start_code、extension_start_code_identifier、quant_matrix_extension_present_flag、load_intra_quantizer_matrix、intra_quantizer_matrix[64]、load_non_intra_quantizer_matrix、non_intra_quantizer_matrix[64]、load_chroma_intra_quantizer_matrix、chroma_intra_quantizer_matrix[64]、load_chroma_non_intra_quantizer_matrix、及びchroma_non_intra_quantizer_matrix[64] である。

【０２８９】extension_start_codeは、この量子化マトリックスエクステンションのスタートを示す開始コードである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。 quant_matrix_extension_present_flagは、この量子化マトリックスエクステンション内のデータエレメントが有効か無効かを示すためのデータである。load_intra_quantizer_matrixは、イントラマクロブロック用の量子化マトリックスデータの存在を示すデータである。intra_quantizer_matrixは、イントラマクロブロック用の量子化マトリックスの値を示すデータである。

【０２９０】load_non_intra_quantizer_matrixは、非イントラマクロブロック用の量子化マトリックスデータの存在を示すデータである。non_intra_quantizer_matrixは、非イントラマクロブロック用の量子化マトリックスの値を表すデータである。load_chroma_intra_quantizer_matrixは、色差イントラマクロブロック用の量子化マトリックス・データの存在を示すデータである。chroma_intra_quantizer_matrixは、色差イントラマクロブロック用の量子化マトリックスの値を示すデータである。load_chroma_non_intra_quantizer_matrixは、色差非イントラマクロブロック用の量子化マトリックス・データの存在を示すデータである。chroma_non_intra_quantizer_matrixは、色差非イントラマクロブロック用の量子化マトリックスの値を示すデータである。

【０２９１】copyright_extension()関数によって定義されるデータエレメントは、図５７に示すように、 extension_start_code、extension_start_code_itentifier、copyright_extension_present_flag、copyright_flag、copyright_identifier、original_or_copy、copyright_number_1、copyright_number_2、及び copyright_number_3から構成される。

【０２９２】extension_start_codeは、コピーライトエクステンションのスタート示す開始コードである。extension_start_code_itentifierどのエクステンションデータが送られるかを示すコードである。 copyright_extension_present_flagは、このコピーライトエクステンション内のデータエレメントが有効か無効かを示すためのデータである。

【０２９３】copyright_flagは、次のコピーライトエクステンション又はシーケンスエンドまで、符号化されたビデオデータに対してコピー権が与えられているか否かを示す。copyright_identifierは、ＩＳＯ／ＩＥＣ ＪＴＣ／ＳＣ２９によって指定されたコピー権の登録機関を識別するためのデータである。original_or_copyは、ビットストリーム中のデータが、オリジナルデータであるかコピーデータであるかを示すデータである。copyright_number_1は、コピーライトナンバーのビット４４から６３を表わすデータである。copyright_number_2は、コピーライトナンバーのビット２２から４３を表わすデータである。copyright_number_3は、コピーライトナンバーのビット０から２１を表わすデータである。

【０２９４】picture_display_extension()関数によって定義されるデータエレメントは、図５８に示すように、extension_start_code_identifier、frame_center_horizontal_offset、frame_center_vertical_offset等である。

【０２９５】extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。 frame_center_horizontal_offsetは、表示エリアの水平方向のオフセットを示すデータであって、number_of_frame_center_offsetsによって定義される数のオフセット値を定義することができる。frame_center_vertical_offsetは、表示エリアを垂直方向のオフセットを示すデータであって、 number_of_frame_center_offsetsによって定義される数のオフセット値を定義することができる。

【０２９６】再び図４７に戻って、extension_and_user_data(2)関数によって定義されるデータエレメントの次には、picture_data()関数によって定義されるデータエレメントが、履歴ストリームとして記述されている。

【０２９７】picture_data()関数によって定義されるデータエレメントは、図５９に示すように、slice()関数によって定義されるデータエレメントである。但し、ビットストリーム中に、slice()関数のスタートコードを示すslice_start_codeが存在しない場合には、このslice()関数によって定義されるデータエレメントはビットストリーム中に記述されていない。

【０２９８】slice()関数は、図６０に示されるように、slice_start_code、slice_quantiser_scale_code、intra_slice_flag、intra_slice、reserved_bits、extra_bit_slice、extra_information_slice、及びextra_bit_slice 等のデータエレメントと、macroblock()関数によって定義されるデータエレメントを、履歴ストリームとして記述するための関数である。

【０２９９】slice_start_codeは、slice()関数によって定義されるデータエレメントのスタートを示すスター

トコードである。slice_quantiser_scale_codeは、このスライス層に存在するマクロブロックに対して設定された量子化ステップサイズを示すデータである。しかし、各マクロブロック毎に、quantiser_scale_codeが設定されている場合には、各マクロブロックに対して設定されたmacroblock_quantiser_scale_codeのデータが優先して使用される。

【０３００】intra_slice_flagは、ビットストリーム中にintra_slice及びreserved_bitsが存在するか否かを示すフラグである。intra_sliceは、スライス層中にノンイントラマクロブロックが存在するか否かを示すデータである。スライス層におけるマクロブロックのいずれかがノンイントラマクロブロックである場合には、intra_sliceは「0」となり、スライス層におけるマクロブロックの全てがノンイントラマクロブロックである場合には、intra_sliceは「1」となる。reserved_bitsは、７ビットのデータであって「0」の値を取る。extra_bit_sliceは、履歴ストリームとして追加の情報が存在することを示すフラグであって、次にextra_information_sliceが存在する場合には「1」に設定される。追加の情報が存在しない場合には「0」に設定される。

【０３０１】これらのデータエレメントの次には、macroblock()関数によって定義されたデータエレメントが、履歴ストリームとして記述されている。

【０３０２】macroblock()関数は、図６１に示すように、macroblock_escape、macroblock_address_increment、及びmacroblock_quantiser_scale_code等のデータエレメントと、macroblock_modes()関数、及び macroblock_vecters(s)関数によって定義されたデータエレメントを記述するための関数である。

【０３０３】macroblock_escapeは、参照マクロブロックと前のマクロブロックとの水平方向の差が３４以上であるか否かを示す固定ビット列である。参照マクロブロックと前のマクロブロックとの水平方向の差が３４以上の場合には、macroblock_address_incrementの値に３３をプラスする。macroblock_address_incrementは、参照マクロブロックと前のマクロブロックとの水平方向の差を示すデータである。もし、このmacroblock_address_incrementの前にmacroblock_escapeが１つ存在するのであれば、このmacroblock_address_incrementの値に３３をプラスした値が、実際の参照マクロブロックと前のマクロブロックとの水平方向の差分を示すデータとなる。

【０３０４】macroblock_quantiser_scale_codeは、各マクロブロック毎に設定された量子化ステップサイズである。各スライス層には、スライス層の量子化ステップサイズを示すslice_quantiser_scale_codeが設定されているが、参照マクロブロックに対してmacroblock_quantiser_scale_codeが設定されている場合には、この量子化ステップサイズを選択する。

【０３０５】macroblock_address_incrementの次には、macroblock_modes()関数によって定義されるデータエレメントが記述されている。macroblock_modes()関数は、図６２に示すように、macroblock_type、frame_motion_type、field_motion_type、dct_type等のデータエレメントを、履歴ストリームとして記述するための関数である。

【０３０６】macroblock_typeは、マクログブロックの符号化タイプを示すデータである。具体的には、図６５乃至図６７に示されるように、macroblock_typeは、macroblock_quant、dct_type_flag、macroblock_motion_forward、及びmacroblock_motion_backwardなどのフラグから生成された可変長データである。 macroblock_quantは、マクロブロックに対して量子化ステップサイズを設定するためのmacroblock_quantiser_scale_codeが設定されているか否かを示すフラグあって、ビットストリーム中にmacroblock_quantiser_scale_codeが存在する場合には、 macroblock_quantは「1」の値を取る。

【０３０７】dct_type_flagは、参照マクロブロックがフレームＤＣＴ又はフィールドＤＣＴで符号化されているかを示すdct_typeが存在するか否かを示すためのフラグ（言い換えるとＤＣＴされているか否かを示すフラグ）であって、ビットストリーム中にdct_typeが存在する場合には、このdct_type_flagは「1」の値を取る。macroblock_motion_forwardは、参照マクロブロックが前方予測されているか否かを示すフラグであって、前方予測されている場合には「1」の値を取る。macroblock_motion_backwardは、参照マクロブロックが後方予測されているか否かを示すフラグであって、後方予測されている場合には「1」の値を取る。

【０３０８】もし、macroblock_motion_forward又はmacroblock_motion_backwardが「1」のときに、ピクチャ構造がフレームのときに、frame_period_frame_dctが「0」のときには、macroblock_typeを表わすデータエレメントの次にframe_motion_typeを表わすデータエレメントが記述されている。尚、このframe_period_frame_dctは、 frame_motion_typeがビットストリーム中に存在するか否かを示すフラグである。

【０３０９】frame_motion_typeは、フレームのマクロブロックの予測タイプを示す２ビットのコードである。予測ベクトルが２個でフィールドベースの予測タイプであれば「００」であって、予測ベクトルが１個でフィールドベースの予測タイプであれば「０１」であって、予測ベクトルが１個でフレームベースの予測タイプであれば「１０」であって、予測ベクトルが１個でディアルプライムの予測タイプであれば「１１」である。

【０３１０】もし、macroblock_motion_forward又はmacroblock_motion_backwardが「1」のときに、ピクチャ構造がフレーム出ない場合には、macroblock_typeを表わすデータエレメントの次にfield_motion_typeを表わすデータエレメントが記述されている。

【０３１１】field_motion_typeは、フィールドのマクロブロックの動き予測を示す２ビットのコードである。予測ベクトルが１個でフィールドベースの予測タイプであれば「０１」であって、予測ベクトルが２個で１８×８マクロブロックベースの予測タイプであれば「１０」であって、予測ベクトルが１個でディアルプライムの予測タイプであれば「１１」である。

【０３１２】もし、ピクチャ構造がフレームで、 frame_period_frame_dctがframe_motion_typeがビットストリーム中に存在することを示し、且つ、frame_period_frame_dctがdct_typeがビットストリーム中に存在することを示す場合には、macroblock_typeを表わすデータエレメントの次にdct_typeを表わすデータエレメントが記述されている。尚、dct_typeは、ＤＣＴがフレームＤＣＴモードか、フィールドＤＣＴモードかを示すデータである。

【０３１３】再び図６１に戻って、もし、参照マクロブロックが前方予測マクロブロックであるか又は参照マクロブロックがイントラマクロブロックであって且つコンシール処理のマクロブロックのいずれかの場合には、motion_vectors(0)関数によって定義されるデータエレメントが記述される。また、参照マクロブロックが後方予測マクロブロックである場合には、motion_vectors(1)関数によって定義されるデータエレメントが記述される。尚、 motion_vectors(0)関数は、第1番めの動きベクトルに関するデータエレメントを記述するための関数であって、motion_vectors(1)関数は、第２番めの動きベクトルに関するデータエレメントを記述するための関数である。

【０３１４】motion_vectors(s)関数は、図６３に示されるように、動きベクトルに関するデータエレメントを記述するための関数である。

【０３１５】もし、動きベクトルが１個でディアルプライム予測モードを使用していない場合には、motion_vertical_field_select[0][s]とmotion_vector(0,s)によって定義されるデータエレメントが記述される。

【０３１６】このmotion_vertical_field_select[r][s]は、第１番目の動きベクトル（前方又は後方のどちらのベクトルであっても良い）が、ボトムフィールドを参照して作られたベクトルであるかトップフィールドを参照して作られたベクトルであるかを示すフラグである。この指標“r”は、第１番めのベクトル又は第２番めのベクトルのいずれのベクトルであるかを示す指標であって、“s”は、予測方向が前方又は後方予測のいずれであるかを示す指標である。

【０３１７】motion_vector(r,s)関数は、図６４に示されるように、motion_code[r][s][t]に関するデータ列と、motion_residual[r][s][t]に関するデータ列と、dmvector[t]を表わすデータとを記述するための関数である。

【０３１８】motion_code[r][s][t]は、動きベクトルの大きさを－１６～＋１６の範囲で表わす可変長のデータである。 motion_residual[r][s][t]は、動きベクトルの残差を表わす可変長のデータである。よって、このmotion_code[r][s][t]と motion_residual[r][s][t]との値によって詳細な動きベクトルを記述することができる。 dmvector[t]は、ディユアルプライム予測モードのときに、一方のフィールド（例えばボトムフィールドに対してトップフィールドを一方のフィールドとする）における動きベクトルを生成するために、時間距離に応じて既存の動きベクトルがスケールされると共に、トップフィールドとボトムフィールドとのライン間の垂直方向のずれを反映させるために垂直方向に対して補正を行うデータである。この指標“r”は、第１番めのベクトル又は第２番めのベクトルのいずれのベクトルであるかを示す指標であって、“s”は、予測方向が前方又は後方予測のいずれであるかを示す指標である。“s”は、動きベクトルが垂直方向の成分であるか水平方向の成分であるかを示すデータである。

【０３１９】図６４に示されるmotion_vector(r,s)関数によって、まず、水平方向のmotion_coder[r][s][0]を表わすデータ列が、履歴ストリームとして記述される。motion_residual[0][s][t]及びmotion_residual[1][s][t]の双方のビット数は、f_code[s][t]で示されるので、 f_code[s][t]が１でない場合には、 motion_residual[r][s][t] がビットストリーム中に存在することを示すことになる。水平方向成分のmotion_residual[r][s][0]が「１」でなくて、水平方向成分のmotion_code[r][s][0]が「０」でないということは、ビットストリーム中にmotion_residual[r][s][0]を表わすデータエレメントが存在し、動きベクトルの水平方向成分が存在するということを意味しているので、その場合には、水平方向成分のmotion_residual[r][s][0]を表わすデータエレメントが記述されている。

【０３２０】続いて、垂直方向のmotion_coder[r][s][1]を表わすデータ列が、履歴ストリームとして記述される。同じようにmotion_residual[0][s][t]及びmotion_residual[1][s][t]の双方のビット数は、f_code[s][t]で示されるので、 f_code[s][t]が１でない場合には、motion_residual[r][s][t] がビットストリーム中に存在することを表わすことになる。motion_residual[r][s][1]が「１」でなくて、motion_code[r][s][1]が「０」でないということは、ビットストリーム中にmotion_residual[r][s][1]を表わすデータエレメントが存在し、動きベクトルの垂直方向成分が存在するということを意味しているので、その場合には、垂直方向成分のmotion_residual[r][s][1]を表わすデータエレメントが記述されている。

【０３２１】なお、可変長フォーマットにおいては、伝送するビットレートを減少させるために、履歴情報を削

減することができる。

【０３２２】すなわち、macroblock_typeとmotion_vectors()は転送するが、quantiser_scale_codeを転送しない場合には、slice_quantiser_scale_codeを”０００００”とすることで、ビットレートを減少させることができる。

【０３２３】また、macroblock_typeのみ転送し、motion_vectors()、quantiser_scale_code、およびdct_typeを転送しない場合には、macroblock_typeとして、” not coded”を使用することで、ビットレートを減少することができる。

【０３２４】さらにまた、picture_coding_typeのみ転送し、slice()以下の情報は全て転送しない場合には、slice_start_codeを持たないpicture_data()を使用することで、ビットレートを減少させることができる。

【０３２５】以上においては、user_data内の２３ビットの連続する”０”が出ないようにする場合に、２２ビット毎に”１”を挿入するようにしたが、２２ビット毎でなくてもよい。また、連続する”０”の個数を数えて”１”を挿入するのではなく、Byte_allignを調べて挿入するようにすることも可能である。

【０３２６】さらに、MPEGにおいては、２３ビットの連続する”０”の発生を禁止しているが、実際には、バイトの先頭から２３ビット連続する場合だけが問題とされ、バイトの先頭ではなく、途中から0が２３ビット連続する場合は、問題とされない。従って、例えば２４ビット毎に、LSB以外の位置に”１”を挿入するようにしてもよい。

【０３２７】また、以上においては、履歴情報を、video elementary streamに近い形式にしたが、packetized elementary streamやtransport streamに近い形式にしてもよい。また、Elementary Streamのuser_dataの場所を、picture_dataの前としたが、他の場所にすることもできる。

【０３２８】なお、上記各処理を行うコンピュータプログラムは、磁気ディスク、CD-ROM等の情報記録媒体よりなる提供媒体のほか、インターネット、デジタル衛星などのネットワーク提供媒体を介してユーザに提供することができる。

【０３２９】

【発明の効果】以上の如く請求項１に記載の復号装置、請求項２に記載の復号方法、および請求項３に記載の提供媒体によれば、ビットストリームのピクチャ層のユーザデータエリアに挿入された符号化履歴情報を復号するようにしたので、小さい規模の装置で、再符号化に伴う画像の劣化を抑制することが可能となる。

【図面の簡単な説明】

【図１】高効率符号化の原理を説明する図である。

【図２】画像データを圧縮する場合におけるピクチャタイプを説明する図である。

【図３】画像データを圧縮する場合におけるピクチャタイプを説明する図である。

【図４】動画像信号を符号化する原理を説明した図である。

【図５】動画像信号を符号化し、復号する装置の構成を示すブロック図である。

【図６】フォーマット変換を説明する図である。

【図７】図５のエンコーダ１８の構成を示すブロック図である。

【図８】図７の予測モード切換回路５２の動作を説明する図である。

【図９】図７の予測モード切換回路５２の動作を説明する図である。

【図１０】図７の予測モード切換回路５２の動作を説明する図である。

【図１１】図７の予測モード切換回路５２の動作を説明する図である。

【図１２】図５のデコーダ３１の構成を示すブロック図である。

【図１３】ピクチャタイプに対応したSNR制御を説明する図である。

【図１４】本発明を適用したトランスコーダ１０１の構成を示すブロック図である。

【図１５】図１４のトランスコーダ１０１のより詳細な構成を示すブロック図である。

【図１６】図１４の復号装置１０２に内蔵されるデコーダ１１１の構成を示すブロック図である。

【図１７】マクロブロックの画素を説明する図である。

【図１８】符号化パラメータが記録される領域を説明する図である。

【図１９】図１４の符号化装置１０６に内蔵されるエンコーダ１２１の構成を示すブロック図である。

【図２０】図１５のヒストリーフォマッタ２１１の構成例を示すブロック図である。

【図２１】図１５のヒストリーデコーダ２０３の構成例を示すブロック図である。

【図２２】図１５のコンバータ２１２の構成例を示すブロック図である。

【図２３】図２２のスタッフ回路３２３の構成例を示すブロック図である。

【図２４】図２２のコンバータ２１２の動作を説明するタイミングチャートである。

【図２５】図１５のコンバータ２０２の構成例を示すブロック図である。

【図２６】図２５のディリート回路３４３の構成例を示すブロック図である。

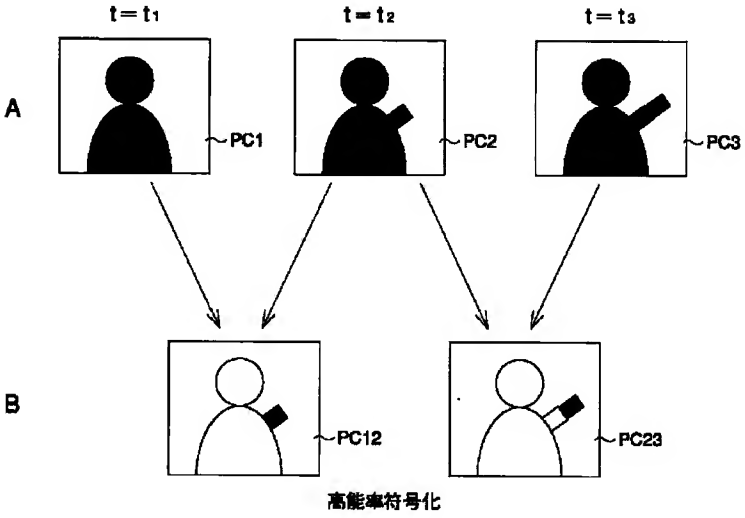
【図２７】図１５のコンバータ２１２の他の構成例を示すブロック図である。

【図２８】図１５のコンバータ２０２の他の構成例を示すブロック図である。

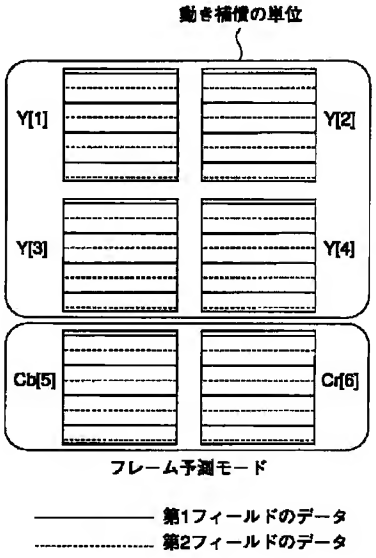
【図２９】図１５のユーザデータフォーマッタ２１３の構成例を示すブロック図である。
【図３０】図１４のトランスコーダ１０１が実際に使用される状態を示す図である。
【図３１】符号化パラメータが記録される領域を説明する図である。
【図３２】図１４の符号化装置１０６の変更可能ピクチャタイプ判定処理を説明するフローチャートである。
【図３３】ピクチャタイプが変更される例を示す図である。
【図３４】ピクチャタイプが変更される他の例を示す図である。
【図３５】図１４の符号化装置１０６の量子化制御処理を説明する図である。
【図３６】図１４の符号化装置１０６の量子化制御処理を説明するフローチャートである。
【図３７】密結合されたトランスコーダ１０１の構成を示すブロック図である。
【図３８】MPEGストリームのシンタックスを説明する図である。
【図３９】図３８のシンタックスの構成を説明する図である。
【図４０】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。
【図４１】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。
【図４２】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。
【図４３】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。
【図４４】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。
【図４５】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。
【図４６】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。
【図４７】可変長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。
【図４８】sequence_header()のシンタックスを説明する図である。
【図４９】sequence_extension()のシンタックスを説明する図である。
【図５０】extension_and_user_data()のシンタックスを説明する図である。
【図５１】user_data()のシンタックスを説明する図である。
【図５２】group_of_pictures_header()のシンタックスを説明する図である。
【図５３】picture_header()のシンタックスを説明する図である。
【図５４】picture_coding_extension()のシンタックスを説明する図である。
【図５５】extension_data()のシンタックスを説明する図である。
【図５６】quant_matrix_extension()のシンタックスを説明する図である。
【図５７】copyright_extension()のシンタックスを説明する図である。
【図５８】picture_display_extension()のシンタックスを説明する図である。
【図５９】picture_data()のシンタックスを説明する図である。
【図６０】slice()のシンタックスを説明する図である。
【図６１】macroblock()のシンタックスを説明する図である。
【図６２】macroblock_modes()のシンタックスを説明する図である。
【図６３】motion_vectors(s)のシンタックスを説明する図である。
【図６４】motion_vector(r,s)のシンタックスを説明する図である。
【図６５】Ｉピクチャに対するmacroblock_typeの可変長符号を説明する図である。
【図６６】Ｐピクチャに対するmacroblock_typeの可変長符号を説明する図である。
【図６７】Ｂピクチャに対するmacroblock_typeの可変長符号を説明する図である。
【図６８】従来のトランスコーダ１３１の構成の一例を示すブロック図である。
【図６９】従来のトランスコーダ１３１の構成の一例を示すブロック図である。
【図７０】従来の符号化装置と復号装置の配置を説明する図である。
【符号の説明】
１　符号化装置，　２　復号化装置，　３　記録媒体，　１２，１３　A/D変換器，　１４　フレームメモリ，　１５　輝度信号フレームメモリ，　１６　色差信号フレームメモリ，　１７　フォーマット変換回路，　１８　エンコーダ，３１　デコーダ，　３２　フォーマット変換回路，　３３　フレームメモリ，３４　輝度信号フレームメモリ，　３５　色差信号フレームメモリ，　３６，３７　D/A変換器，　５０　動きベクトル検出回路，　５１　フレームメモリ，５２　予測モード切り替え回路，　５３　演算部，　５４　予測判定回路，５５　DCTモード切り替え回路，　５６　DCT回路，　５７　量子化回路，　５８可変長符号化回路，　５９　送信バッファ，　６０　逆量子化回路，　６１IDCT回路，　６２　演算器，　６３　フレームメモリ，　６４　動き補償回路，８１　受信バッファ，　８２　可変長復号化回

路，　８３　逆量子化回路，８４　IDCT回路，　８５　演算器，　８６　フレームメモリ，　８７　動き補償回路，　１０１　トランスコーダ，　１０２　復号装置，　１０３　符号化パラメータ多重装置，　１０５　符号化パラメータ分離装置，　１０６　符号化装置，　１０６　SDTI，　１１１　デコーダ，　１１２　可変長復号化回路，　１２１　エンコーダ，　１２２　符号化パラメータコントローラ，　１３１　トランスコーダ，　１３２　復号装置，　１３３　符号化装置，　１３４　動き検出部，　１３５　符号化部

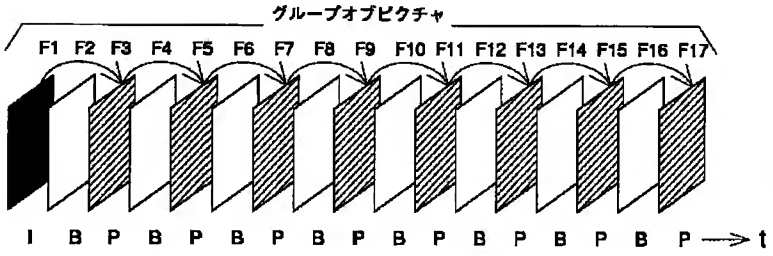
【図１】



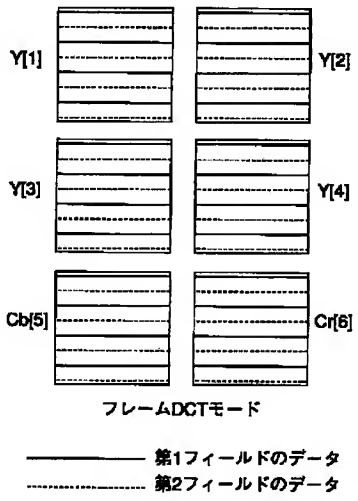
【図８】



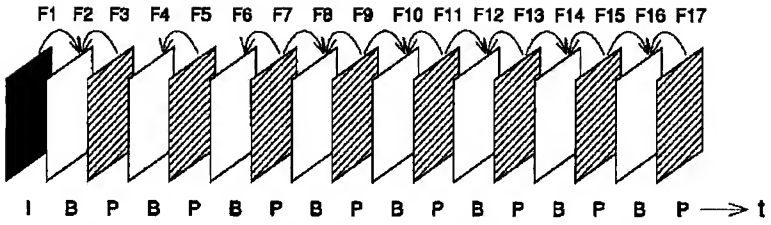
【図２】



【図１０】



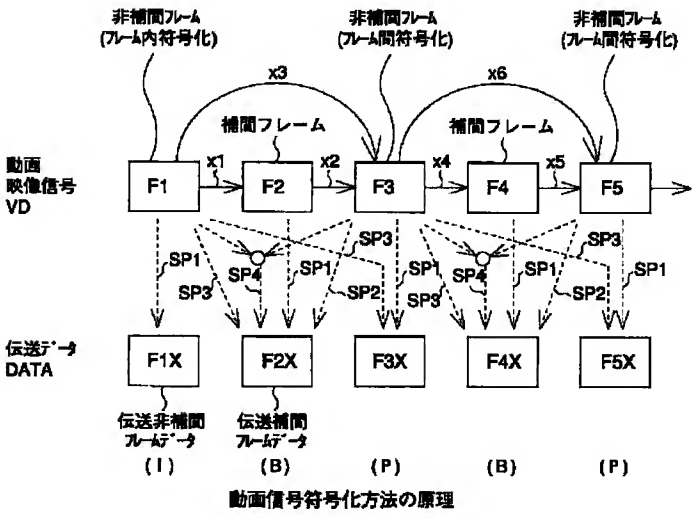
【図３】



両方向予測 B-ピクチャ

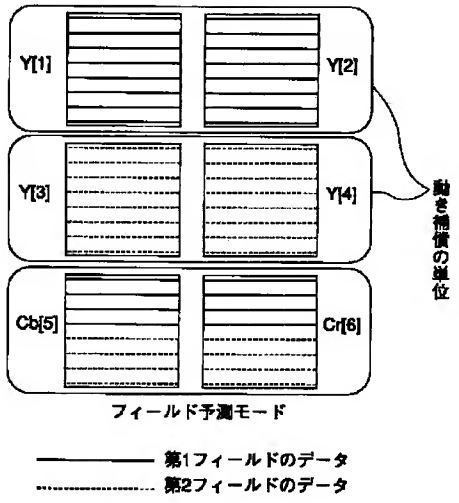
ピクチャタイプ I,P,B-picture

【図４】



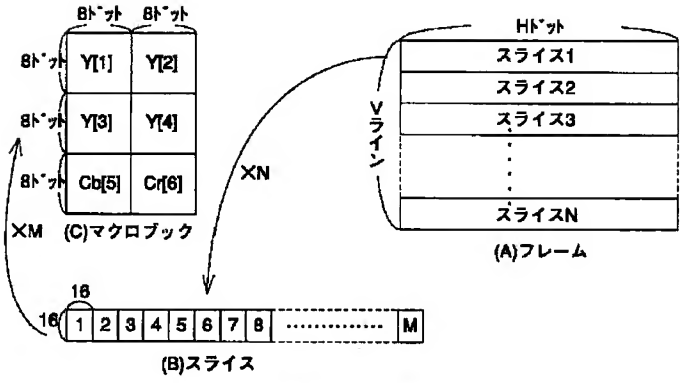
動画信号符号化方法の原理

【図９】



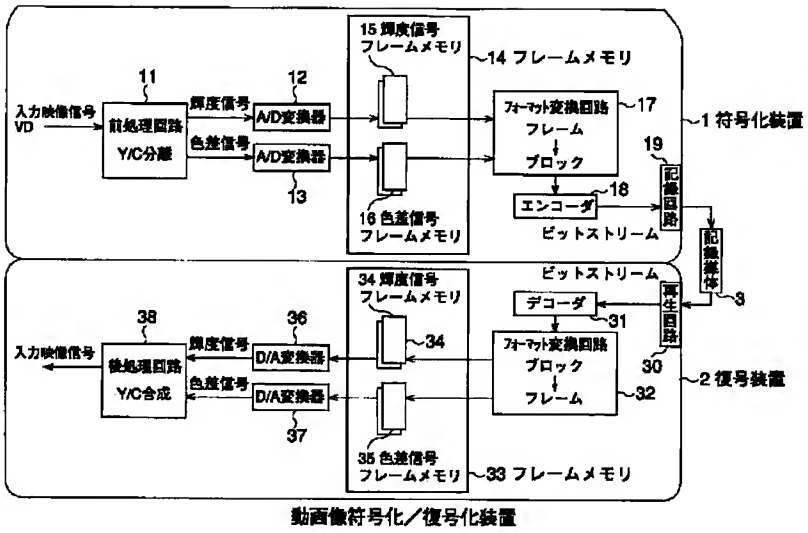
フレーム／フィールド予測モード

【図６】



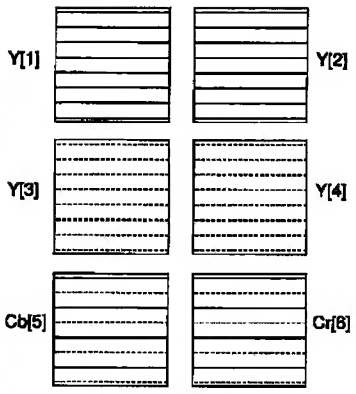
画像データの構造

【図５】



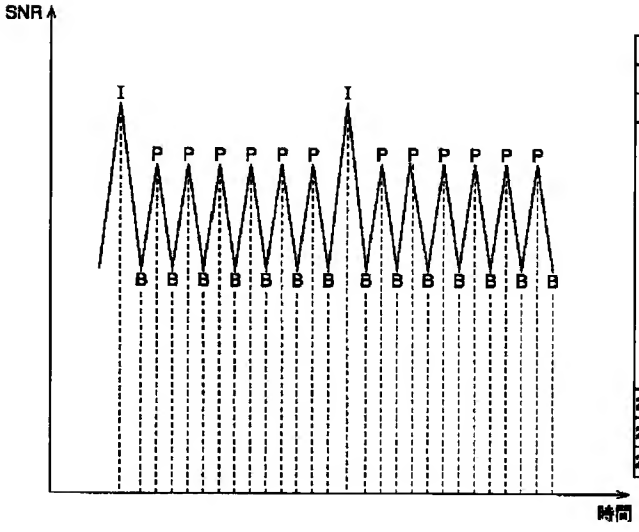
動画像符号化／復号化装置

【図１１】

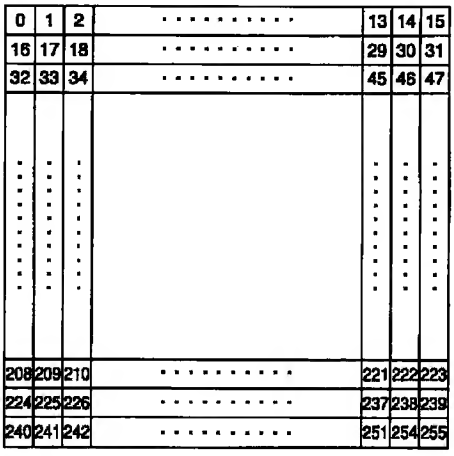


フレーム／フィールドDCTモード

【図１３】

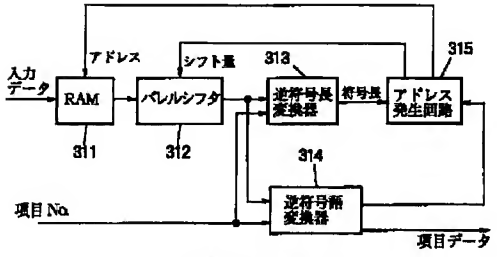


【図１７】



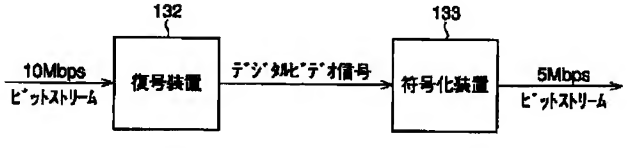
マクロブロック

【図２１】



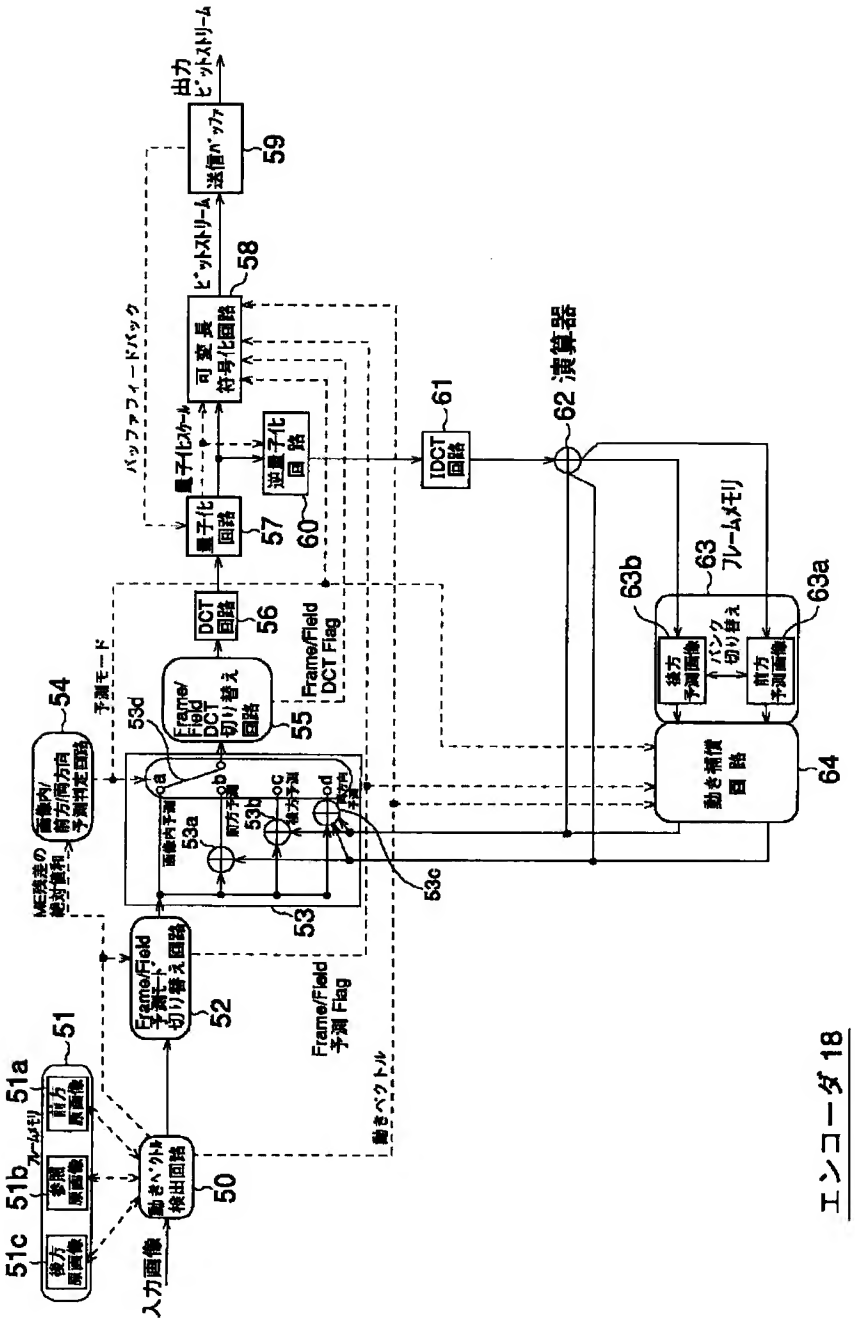
ヒストリデコーダ 203

【図６８】

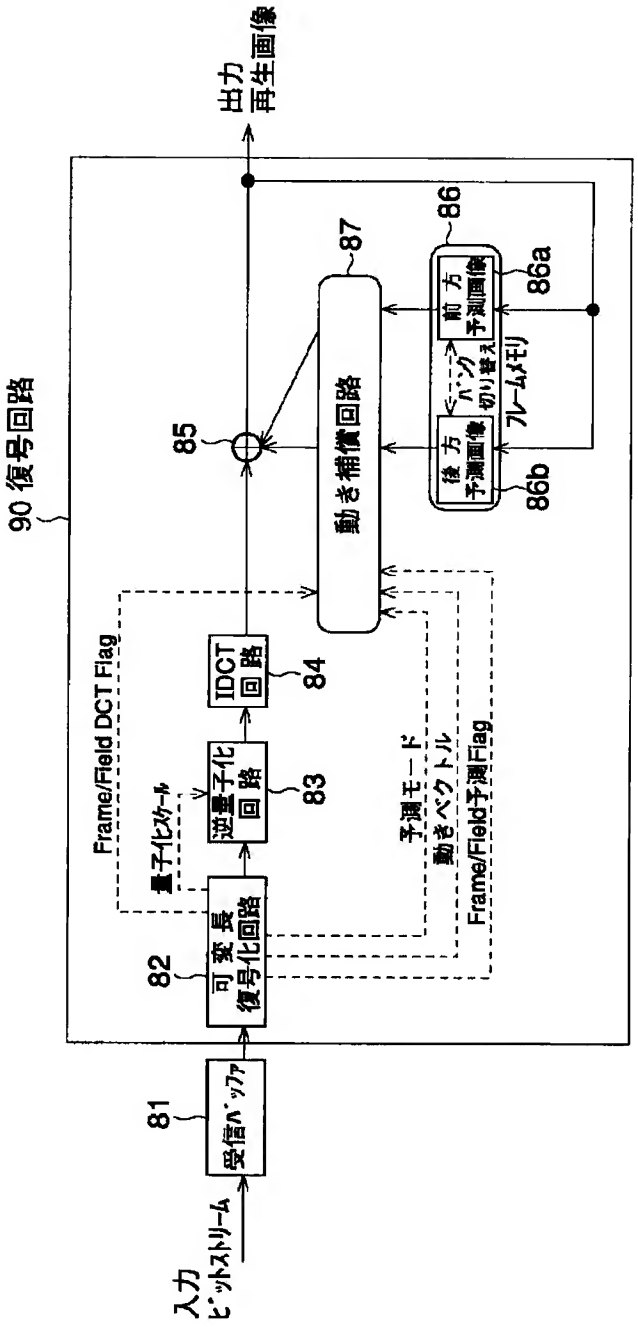


トランスコーダ 131

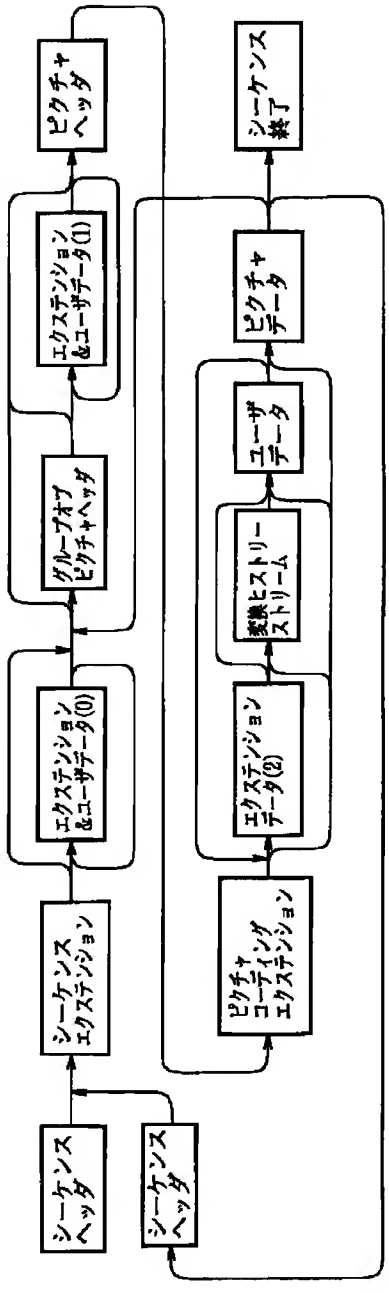
【図７】


エンコーダ 18
入力画像
動きベクトル検出回路 50
フレームメモリ 51
51a
51b
51c
前方原画像
参照原画像
後方原画像
Frame/Field 予測モード切り替え回路 52
53
53a
53b
53c
53d
画像内予測
前方予測
後方予測
両方向予測
Frame/Field DCT 切り替え回路 55
DCT 回路 56
量子化回路 57
可変長符号化回路 58
送信バッファ 59
逆量子化回路 60
IDCT 回路 61
62 演算器
63 フレームメモリ
63a
63b
前方予測画像
バンク切り替え
後方予測画像
動き補償回路 64
54
画像内/前方/両方向予測判定回路
ME残差の絶対値和
予測モード
Frame/Field DCT Flag
Frame/Field 予測モード Flag
動きベクトル
量子化ステップ
ビットストリーム
出力ビットストリーム
バックフィードバック

【図１２】

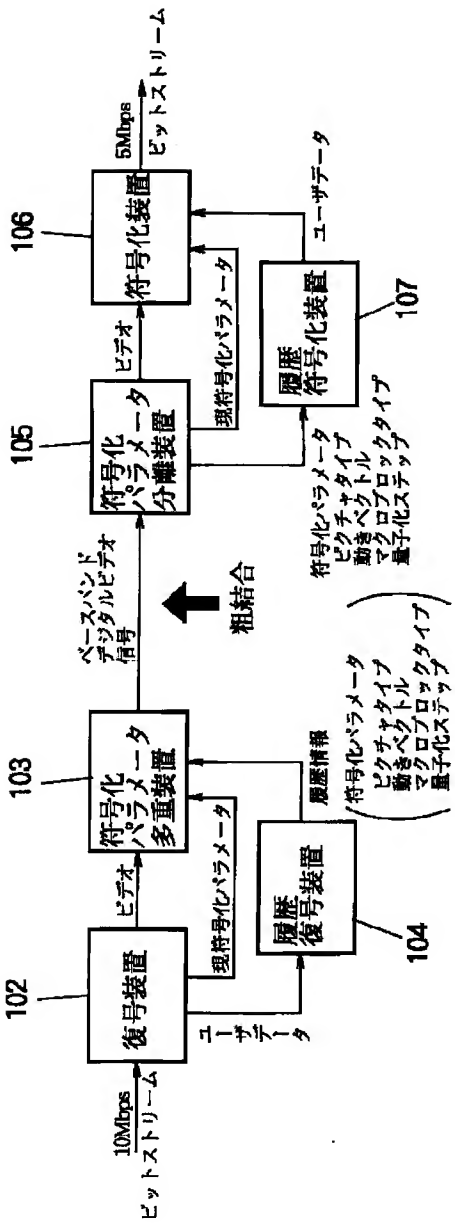


【図３９】



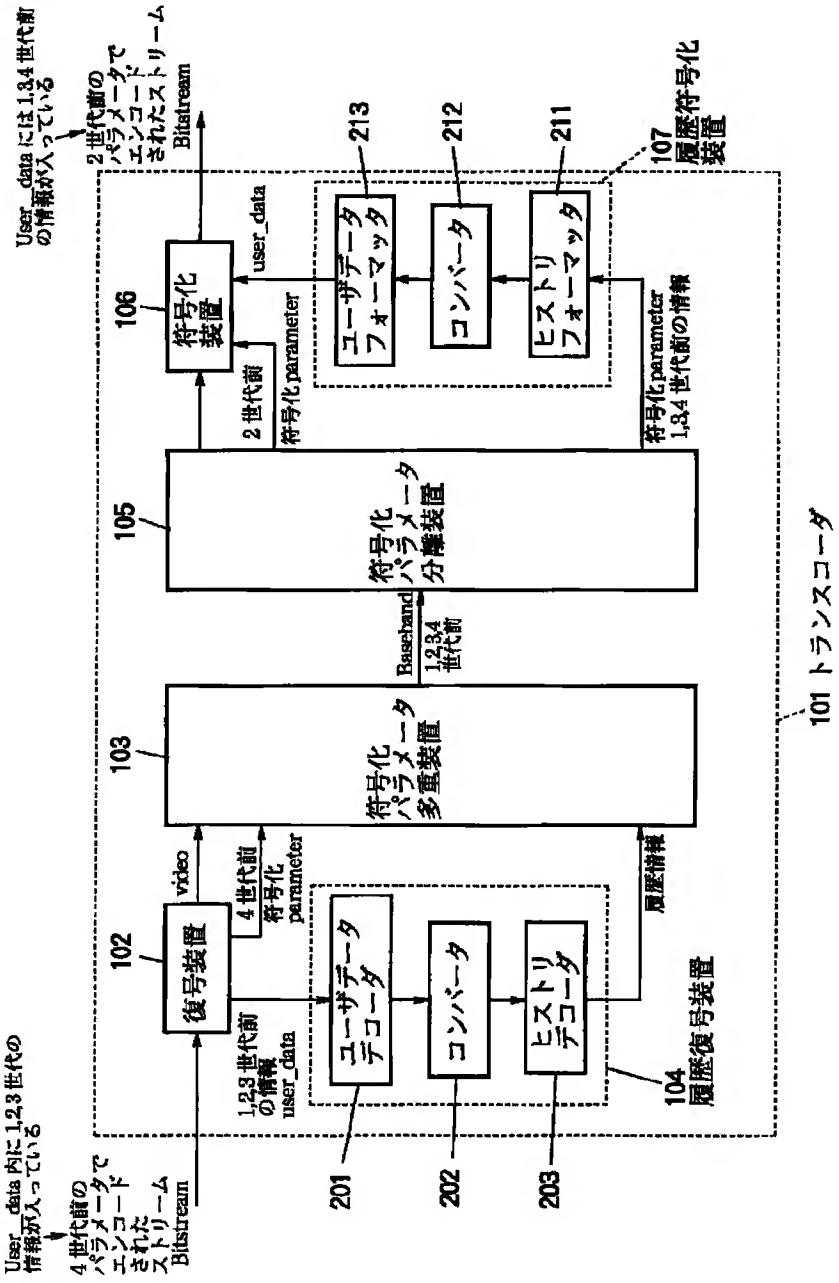
デコーダ 31

【図１４】

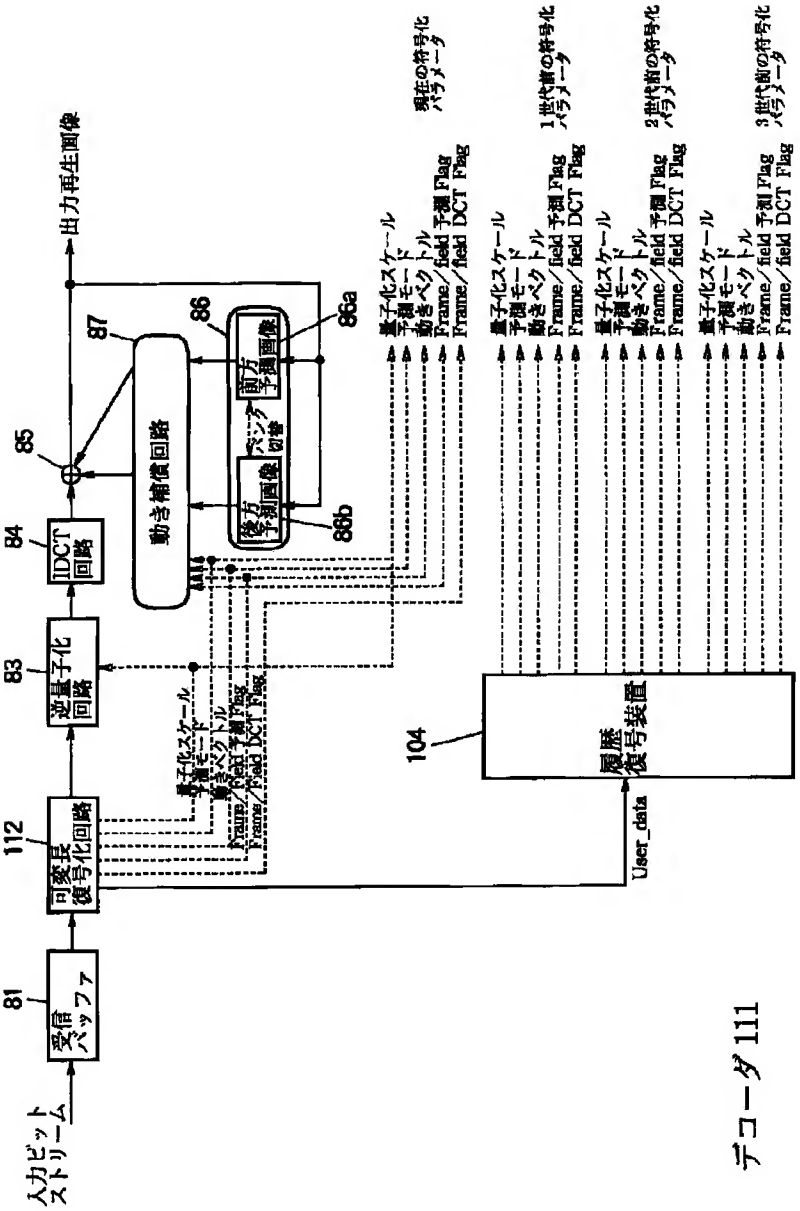


トランスコーダ 101

【図１５】


User_data 内に 1,2,3 世代の情報が入っている
4 世代前のパラメータでエンコードされたストリーム Bitstream
User_data には 1,3,4 世代前の情報が入っている
2 世代前のパラメータでエンコードされたストリーム Bitstream
101 トランスコーダ
102 復号装置
video
4 世代前 符号化 parameter
1,2,3 世代前の情報 user_data
103 符号化パラメータ多重装置
Baseband 1,2,3,4 世代前
105 符号化パラメータ分離装置
106 符号化装置
user_data
2 世代前 符号化 parameter
201 ユーザデータデコーダ
202 コンバータ
203 ヒストリデコーダ
履歴情報
104 履歴復号装置
213 ユーザデータフォーマッタ
212 コンバータ
211 ヒストリフォーマッタ
107 履歴符号化装置
符号化 parameter 1,3,4 世代前の情報

【図１６】


入力ビットストリーム
81
受信バッファ
112
可変長復号化回路
83
逆量子化回路
84
IDCT回路
85
動き補償回路
87
86
86a
86b
前方予測画像
バック予測画像
後方予測画像
出力再生画像
104
履歴復号装置
User_data
量子化スケール
予測モード
動きベクトル
Frame/field 予測 Flag
Frame/field DCT Flag
現在の符号化パラメータ
1世代前の符号化パラメータ
2世代前の符号化パラメータ
3世代前の符号化パラメータ

デコーダ 111

【図１８】

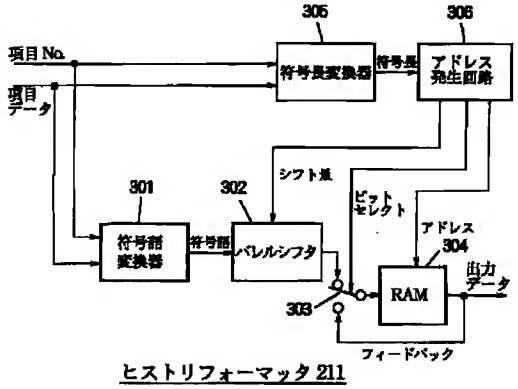
区間

D9	Cb[0][9]	Y[0][9]	Cr[0][9]	Y[1][9]	Cb[1][9]	Y[2][9]	Cr[1][9]	Y[3][9]
D8	Cb[0][8]	Y[0][8]	Cr[0][8]	Y[1][8]	Cb[1][8]	Y[2][8]	Cr[1][8]	Y[3][8]
D7	Cb[0][7]	Y[0][7]	Cr[0][7]	Y[1][7]	Cb[1][7]	Y[2][7]	Cr[1][7]	Y[3][7]
D6	Cb[0][6]	Y[0][6]	Cr[0][6]	Y[1][6]	Cb[1][6]	Y[2][6]	Cr[1][6]	Y[3][6]
D5	Cb[0][5]	Y[0][5]	Cr[0][5]	Y[1][5]	Cb[1][5]	Y[2][5]	Cr[1][5]	Y[3][5]
D4	Cb[0][4]	Y[0][4]	Cr[0][4]	Y[1][4]	Cb[1][4]	Y[2][4]	Cr[1][4]	Y[3][4]
D3	Cb[0][3]	Y[0][3]	Cr[0][3]	Y[1][3]	Cb[1][3]	Y[2][3]	Cr[1][3]	Y[3][3]
D2	Cb[0][2]	Y[0][2]	Cr[0][2]	Y[1][2]	Cb[1][2]	Y[2][2]	Cr[1][2]	Y[3][2]
D1	1世代前		2世代前		3世代前		最新	
D0								
	Cb[0][x]	Y[0][x]	Cr[0][x]	Y[1][x]	Cb[1][x]	Y[2][x]	Cr[1][x]	Y[3][x]

画像データ領域

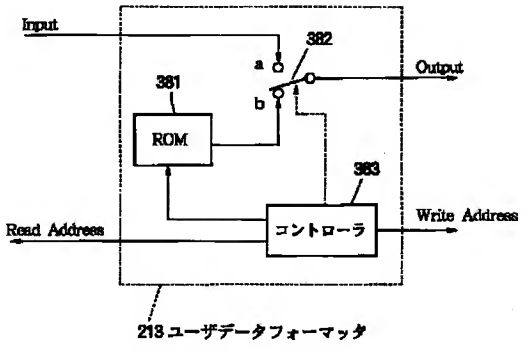
符号化パラメータ領域

【図２０】



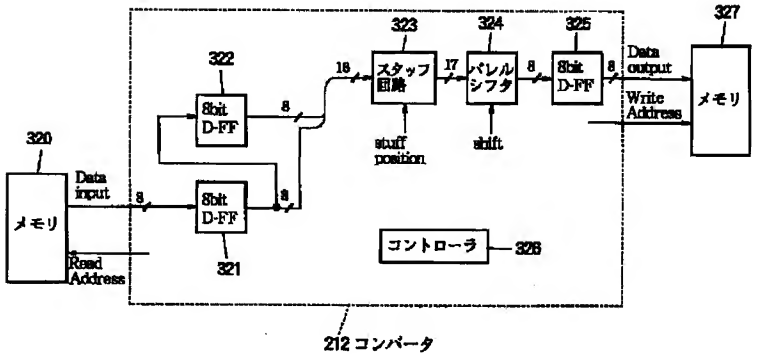
ヒストリフォーマッタ 211

【図２９】



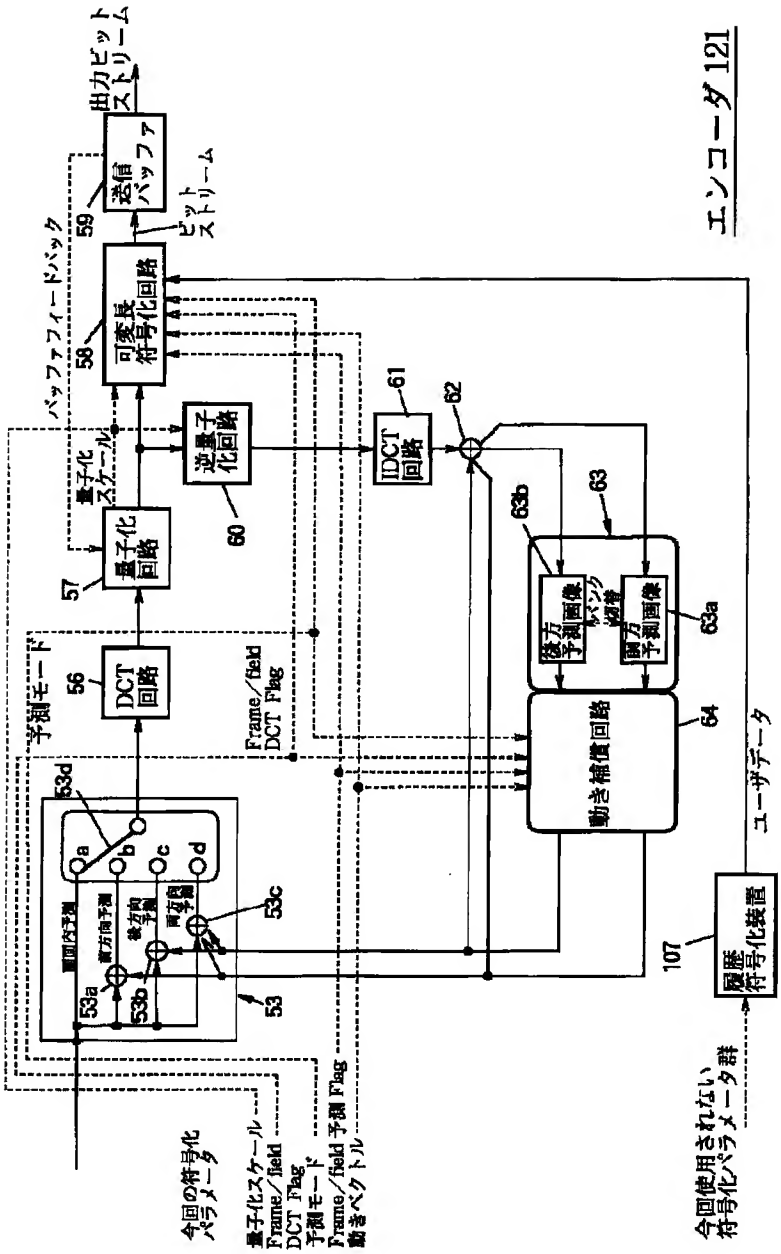
213 ユーザデータフォーマッタ

【図２２】

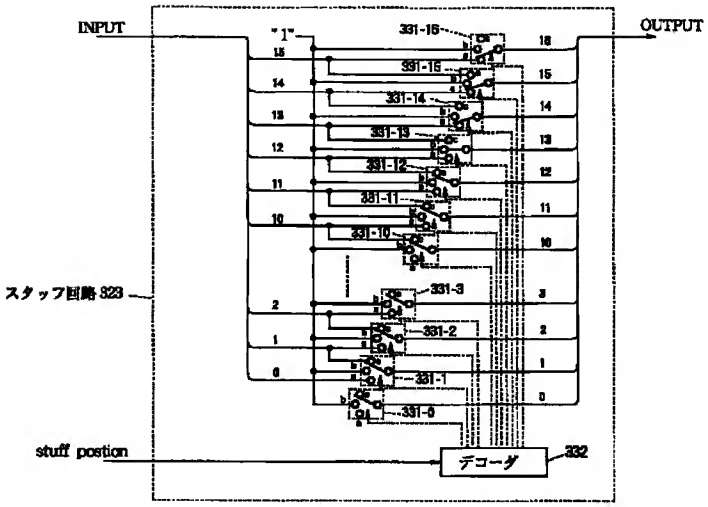


212 コンパータ

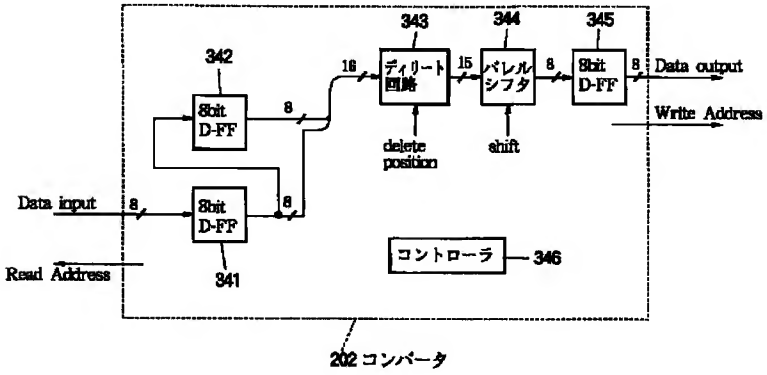
【図１９】


エンコーダ121
出力ビットストリーム
送信バッファ
59
ビットストリーム
バッファフィードバック
可変長符号化回路
58
量子化スケール
量子化回路
57
逆量子化回路
60
IDCT回路
61
62
63
63b
63a
両方予測画像
バイアス値
前方予測画像
後方予測画像
動き補償回路
64
予測モード
DCT回路
56
Frame/field DCT Flag
53d
a
b
c
d
53c
53a
53b
53
前回値予測
後方向予測
両方向予測
履歴符号化装置
107
ユーザデータ
今回の符号化パラメータ
量子化スケール
Frame/field
DCT Flag
予測モード
Frame/field 予測 Flag
動きベクトル
今回使用されない符号化パラメータ群

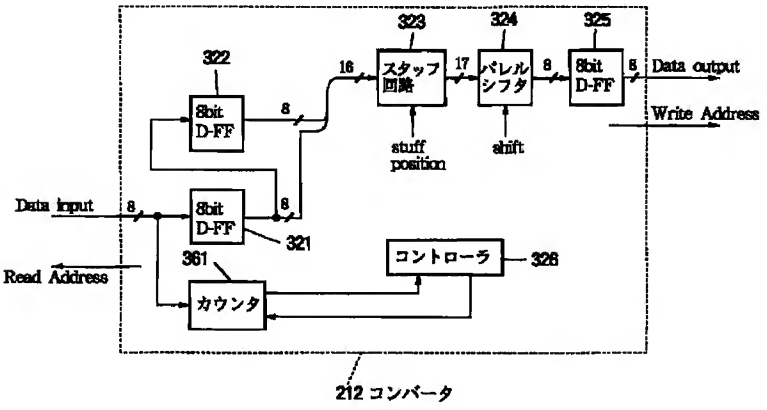
【図２３】


INPUT
OUTPUT
"1"
331-15
331-14
331-13
331-12
331-11
331-10
331-3
331-2
331-1
331-0
スタッフ回路 323
stuff postion
デコーダ
332

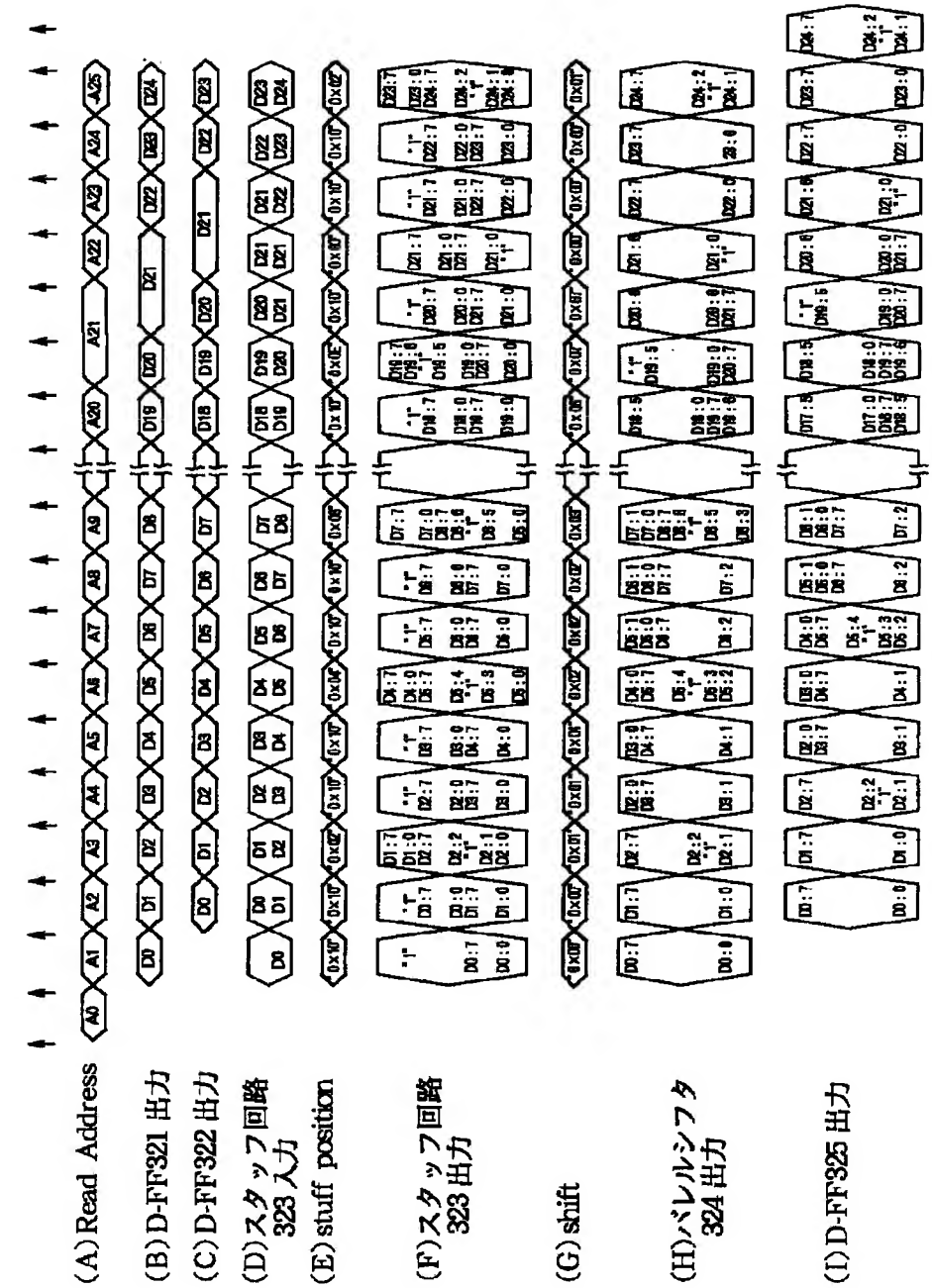
【図２５】


342
343
344
345
8bit D-FF
ディリート回路
バレルシフタ
8bit D-FF
Data output
Write Address
delete position
shift
Data input
Read Address
コントローラ
346
341
202 コンバータ

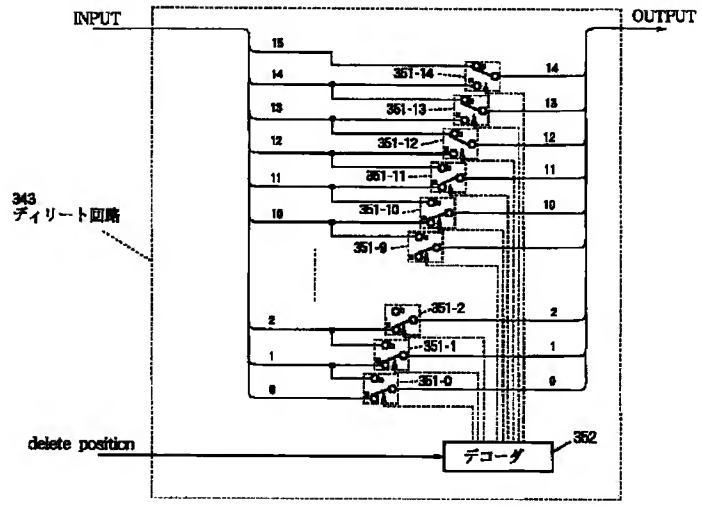
【図２７】


322
323
324
325
8bit D-FF
スタッフ回路
バレルシフタ
8bit D-FF
Data output
Write Address
stuff position
shift
Data input
Read Address
321
361
カウンタ
コントローラ
326
212 コンバータ

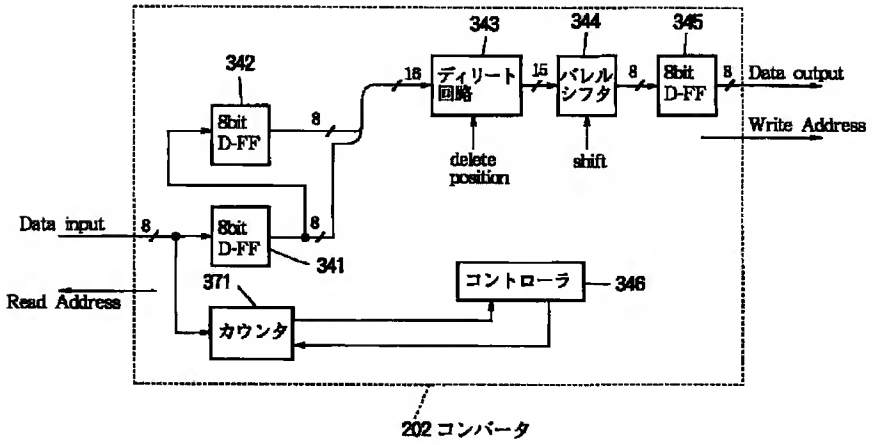
【図２４】


(A) Read Address
(B) D-FF321 出力
(C) D-FF322 出力
(D) スタッフ回路 323 入力
(E) stuff position
(F) スタッフ回路 323 出力
(G) shift
(H) パレルシフタ 324 出力
(I) D-FF325 出力

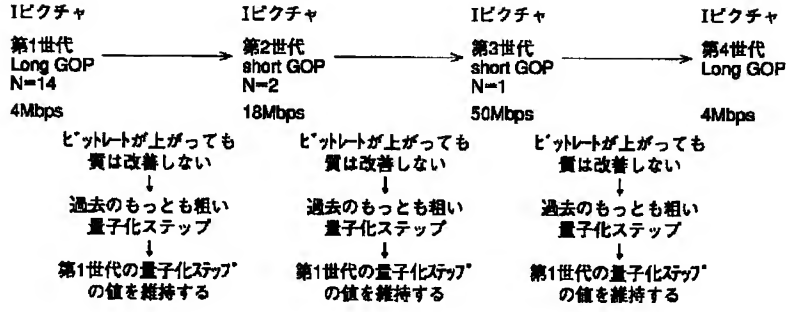
【図２６】


INPUT
OUTPUT
343
ディリート回路
351-14
351-13
351-12
351-11
351-10
351-9
351-2
351-1
351-0
delete position
デコーダ
352

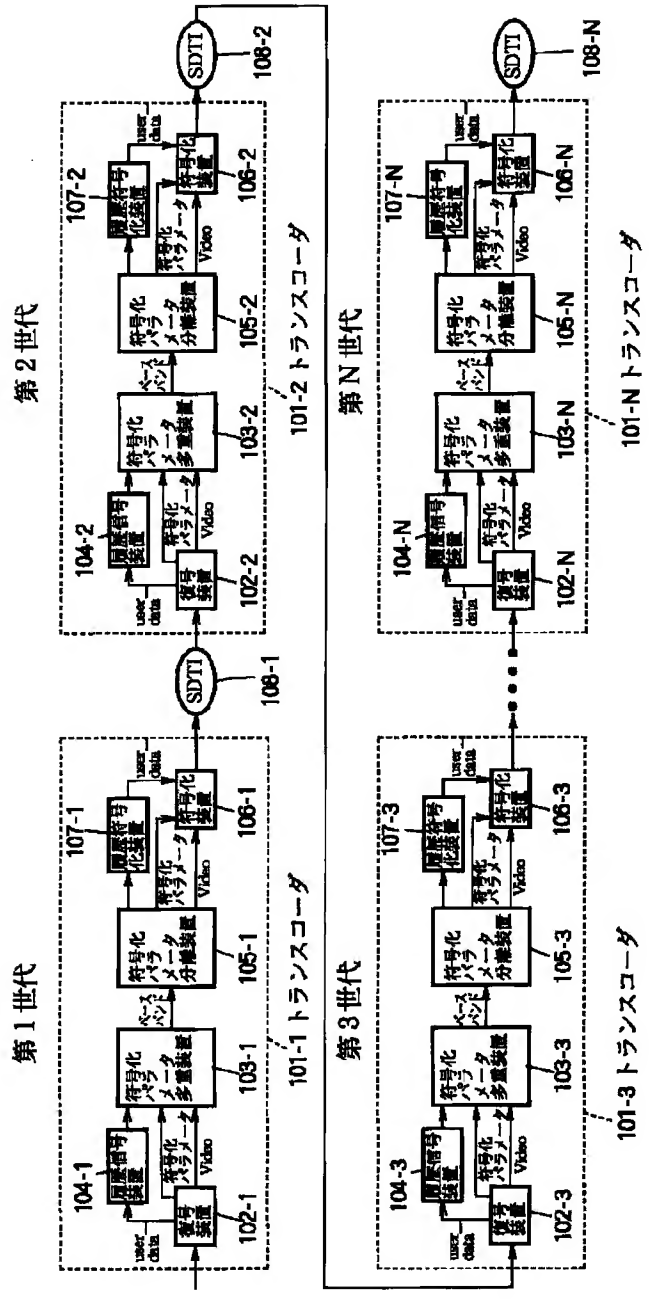
【図２８】


342
8bit D-FF
343
ディリート回路
344
バレルシフタ
345
8bit D-FF
Data output
Write Address
delete position
shift
Data input
8bit D-FF
341
371
Read Address
カウンタ
コントローラ
346
202 コンバータ

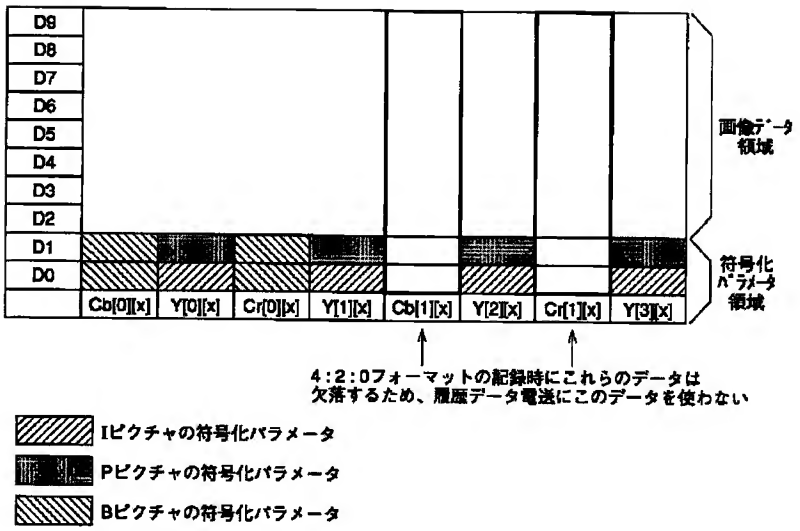
【図３５】


Iピクチャ
第1世代
Long GOP
N=14
4Mbps
Iピクチャ
第2世代
short GOP
N=2
18Mbps
Iピクチャ
第3世代
short GOP
N=1
50Mbps
Iピクチャ
第4世代
Long GOP
4Mbps
ﾋﾞｯﾄﾚｰﾄが上がっても
質は改善しない
過去のもっとも粗い
量子化ステップ
第1世代の量子化ｽﾃｯﾌﾟ
の値を維持する

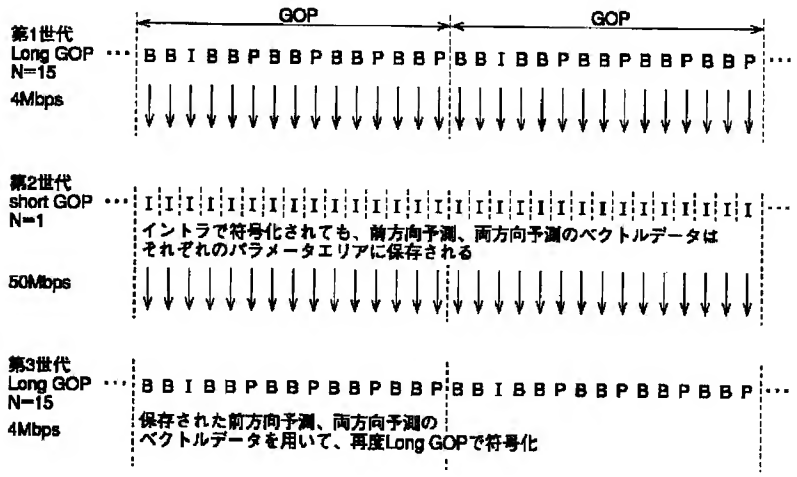
【図３０】


第１世代
第２世代
第３世代
第Ｎ世代
101-1 トランスコーダ
101-2 トランスコーダ
101-3 トランスコーダ
101-N トランスコーダ
SDTI
102-1
102-2
102-3
102-N
103-1
103-2
103-3
103-N
104-1
104-2
104-3
104-N
105-1
105-2
105-3
105-N
106-1
106-2
106-3
106-N
107-1
107-2
107-3
107-N
108-1
108-2
108-N
復号装置
周波符号装置
符号化パラメータ
符号化装置
符号化パラメータ多重装置
ベースバンド
Video
user data

【図３１】



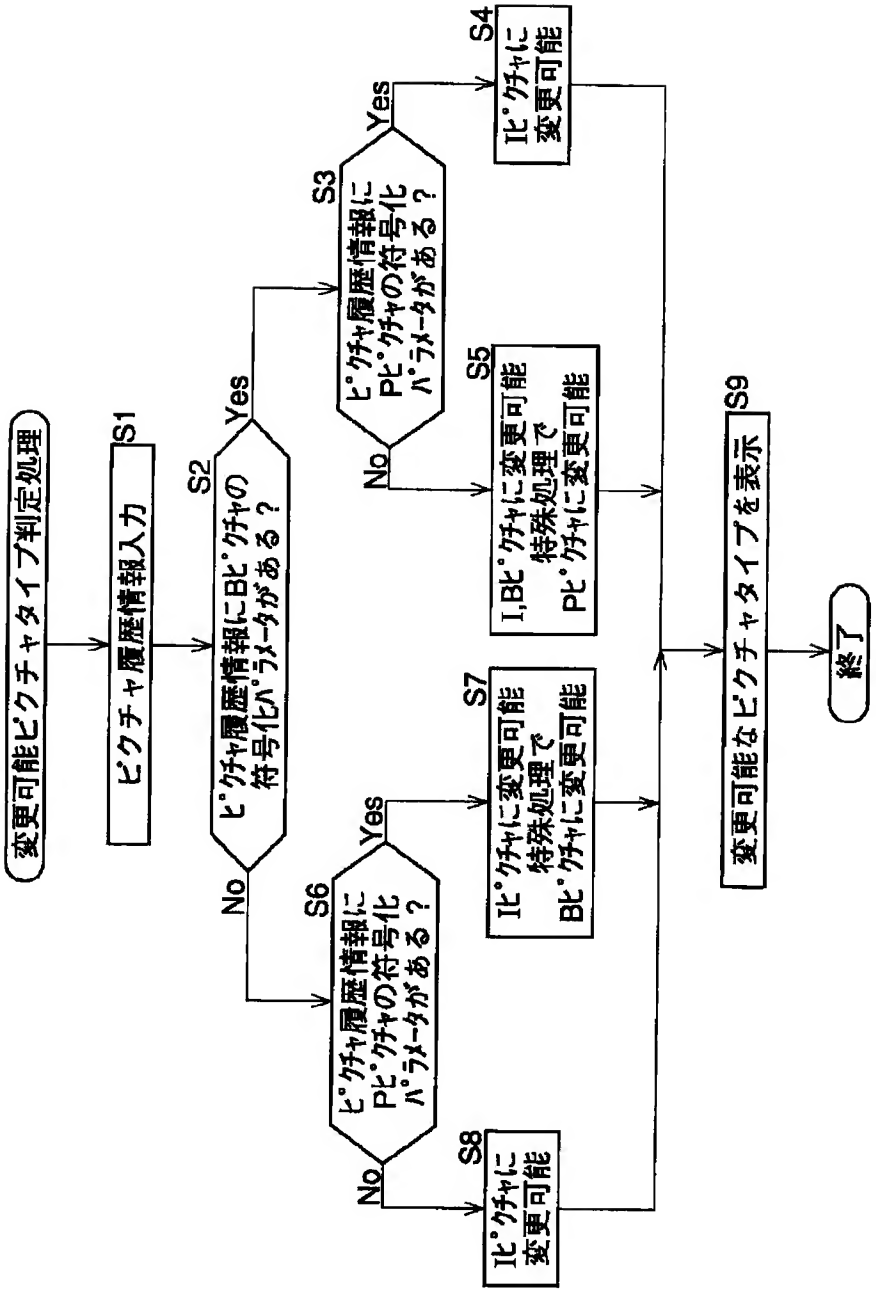
【図３３】



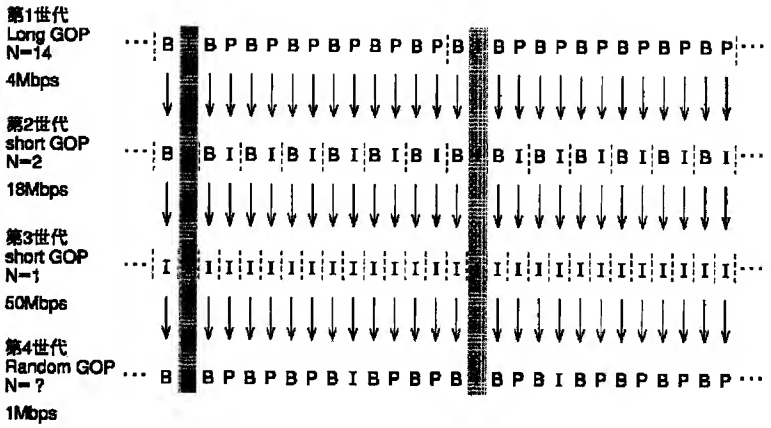
【図５０】

extension_and_user_data(i) {	No. of bits	Mnemonic
while ((nextbits()== extension_start_code) ||		
(nextbits()== user_data_start_code)) {		
if ((i == 2) && (nextbits()== extension_start_code))		
extension_data()		
if (nextbits()== user_data_start_code)		
user_data()		
}		
}		

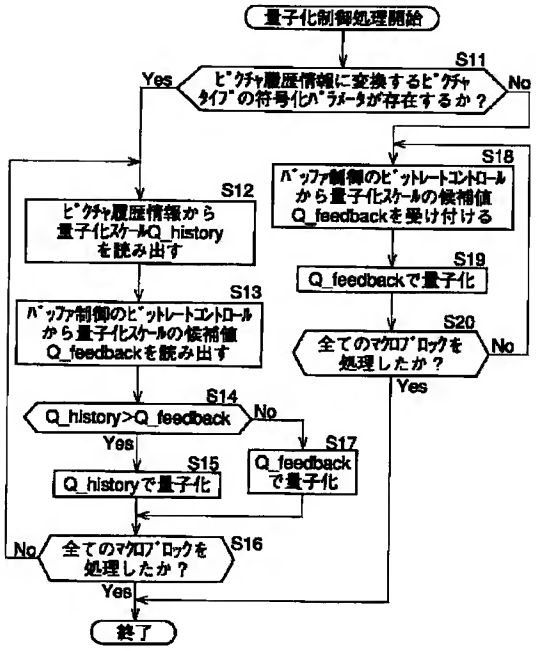
【図３２】


変更可能ピクチャタイプ判定処理
S1
ピクチャ履歴情報入力
S2
ピクチャ履歴情報にBピクチャの符号化パラメータがある？
Yes
No
S3
ピクチャ履歴情報にPピクチャの符号化パラメータがある？
Yes
No
S4
Iピクチャに変更可能
S5
I,Bピクチャに変更可能
特殊処理で
Pピクチャに変更可能
S6
ピクチャ履歴情報にPピクチャの符号化パラメータがある？
Yes
No
S7
Iピクチャに変更可能
特殊処理で
Bピクチャに変更可能
S8
Iピクチャに変更可能
S9
変更可能なピクチャタイプを表示
終了

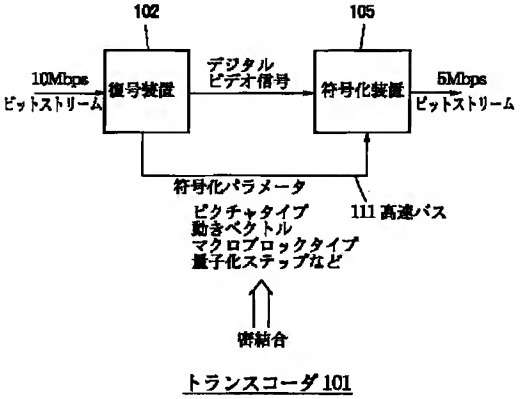
【図３４】



【図３６】



【図３７】



【図５１】

user_data() {	No. of bits	Mnemonic
user_data_start_code	32	bslbf
while(nextbits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	uimsbf
}		
next_start_code()		
}		

【図３８】

stream with history data

video_sequence ( ) {	No.of bits	Mnemonic
next_start_code ( )		
sequence_header ( )		
sequence_extension ( )		
do {		
extension_and_user_data (0)		
do {		
if (nextbits ( ) = =group_start_code) {		
group_of_pictures_header (1)		
extension_and_user_data (1)		
}		
picture_header ( )		
picture_coding_extension ( )		
while ((nextbits ( ) = =extension_start_code) ||		
(nextbits ( ) = =user_data_start_code)) {		
if (nextbits ( ) = =extension_start_code)		
extension_data (2)		
if (nextbits ( ) = =user_data_start_code) {		
user_data_start_code	32	bslbf
if (nextbits ( ) = =History_Data_ID) {		
History_Data_ID	8	bslbf
converted_history_stream ( )		
}		
else {		
user_data ( )		
}		
}		
}		
picture_data ( )		
} while ((nextbits ( ) = =picture_start_code) ||		
(nextbits ( ) = =group_start_code))		
if (nextbits ( ) ! =sequence_end_code) {		
sequence_header ( )		
sequence_extension ( )		
}		
} while (nextbits ( ) ! =sequence_end_code)		
sequence_end_code	32	bslbf
}		

【図５９】

picture_data() {	No. of bits	Mnemonic
while (nextbits() == slice_start_code) {		
slice()		
}		
next_start_code()		
}		

【図４０】

history stream(40-1)

history_stream( ){	bits	value
sequence_header		
sequence_header_code	32	000001B3
sequence_header_present_flag	1	
horizontal_size_value	12	
marker_bit	1	1
vertical_size_value	12	
aspect_ratio_information	4	
frame_rate_code	4	
marker_bit	1	1
bit_rate_value	18	
marker_bit	1	1
vbv_buffer_size_value	10	
constrained_parameter_flag	1	0
load_intra_quantiser_matrix	1	
load_non_intra_quantiser_matrix	1	
maker_bits	5	1F
intra_quantiser_matrix[64]	8*64	
non_intra_quantiser_matrix[64]	8*64	
sequence_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	1
sequence_extension_present_flag	1	
profile_and_level_indication	8	
progressive_sequence	1	
chroma_format	2	
horizontal_size_extension	2	
vertical_size_extension	2	
marker_bit	1	1
bit_rate_extension	12	
vbv_buffer_size_extension	8	
low_delay	1	
marker_bit	1	1

【図４１】

history stream(40-2)

	bits	value
frame_rate_extension_n	2	
frame_rate_extension_d	5	
marker_bits	6	3F
sequence_display_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	2
sequence_display_extension_present_flag	1	
video_format	3	
colour_description	1	
colour_primaries	8	
transfer_characteristics	8	
marker_bit	1	1
matrix_coefficients	8	
display_horizontal_size	14	
marker_bit	1	1
display_vertical_size	14	
marker_bit	1	1
macroblock_assignment_in_user_data		
macroblock_assignment_present_flag	1	
marker_bits	7	7F
v_phase	8	
h_phase	8	
group_of_picture_header		
group_start_code	32	000001B8
group_of_picture_header_present_flag	1	
time_code	25	
closed_gop	1	
broken_link	1	
marker_bits	4	F
picture_header		
picture_start_code	32	00000100

【図５２】

group_of_pictures_header() {	No. of bits	Mnemonic
group_start_code	32	bslbf
time_code	25	bslbf
closed_gop	1	uimsbf
broken_link	1	uimsbf
next_start_code()		
}		

【図４２】

history stream(40-3)

	bits	value
temporal_reference	10	
picture_coding_type	3	
marker_bit	1	1
vbv_delay	16	
full_pel_forward_vector	1	
forward_f_code	3	
full_pel_backward_vector	1	
marker_bit	1	1
backward_f_code	3	
marker_bit	1	1
picture_coding_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	8
f_code[0][0]	4	
f_code[0][1]	4	
f_code[1][0]	4	
f_code[1][1]	4	
intra_dc_precision	2	
picture_structure	2	
top_filed_first	1	
frame_pred_frame_dct	1	
concealment_motion_vectors	1	
q_scale_type	1	
marker_bit	1	1
intra_vlc_format	1	
alternate_scan	1	
repeat_first_field	1	
chroma_420_type	1	
progressive_frame	1	
composite_display_flag	1	
v_axis	1	
field_sequence	3	
sub_carrier	1	
burst_amplitude	7	

【図４３】

history stream(40-4)

	bits	value
marker_bit	1	1
sub_carrier_phase	8	
quant_matrix_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	3
quant_matrix_extension_present_flag	1	
load_intra_quantiser_matrix	1	
marker_bits	2	3
intra_quantiser_matrix[64]	8*64	
load_non_intra_quantiser_matrix	1	
marker_bits	7	7F
non_intra_quantiser_matrix[64]	8*64	
load_chroma_intra_quantiser_matrix	1	
marker_bits	7	7F
chorma_intra_quantiser_matrix[64]	8*64	
load_chroma_non_intra_quantiser_matrix	1	
marker_bits	7	7F
chroma_non_intra_quantiser_matrix[64]	8*64	
copyright_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	4
copyright_extension_present_flag	1	
copyright_flag	1	
copyright_identifier	8	
original_or_copy	1	
marker_bit	1	
copyright_number_1	20	
maker_bit	1	
copyright_number_2	22	
maker_bit	1	
copyright_number_3	22	3F
maker_bits	6	

【図４４】

history stream(40-5)

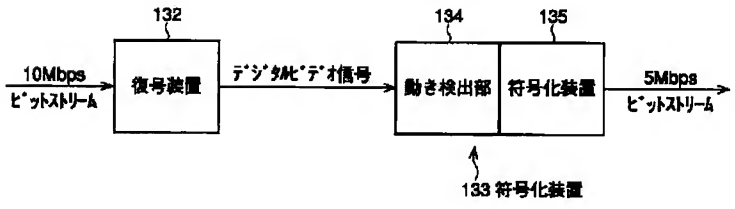
	bits	value
picture_display_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	7
picture_display_extension_present_flag	1	
frame_centre_horizontal_offset_1	16	
marker_bit	1	1
frame_centre_vertical_offset_1	16	
marker_bit	1	1
frame_centre_horizontal_offset_2	16	
marker_bit	1	1
frame_centre_vertical_offset_2	16	
marker_bit	1	1
frame_centre_horizontal_offset_3	16	
marker_bit	1	1
frame_centre_vertical_offset_3	16	
marker_bits	6	3F
user_data		
user_data_start_code	32	000001B2
user_data	2048	
while(macroblock ¡= macroblock_count){		
macroblock		
macroblock_address_h	8	
macroblock_address_v	8	
slice_header_present_flag	1	
skipped_macroblock_flag	1	
marker_bit	1	1
macroblock_modes()		
macroblock_quant	1	
macroblock_motion_forward	1	
macroblock_motion_backward	1	
macroblock_pattern	1	
macroblock_intra	1	

【図４５】

history stream(40-6)

	bits	value
spatial_temporal_weight_code_flag	1	
frame_motion_type	2	
field_motion_type	2	
dct_type	1	
marker_bits	2	3
quantiser_scale_code	5	
marker_bits	3	7
PMV[0][0][0]	14	
marker_bits	2	3
PMV[0][0][1]	14	
motion_vertical_field_select[0][0]	1	
marker_bit	1	1
PMV[0][1][0]	14	
marker_bits	2	3
PMV[0][1][1]	14	
motion_vertical_field_select[0][1]	1	
marker_bit	1	1
PMV[1][0][0]	14	
marker_bits	2	3
PMV[1][0][1]	14	
motion_vertical_field_select[1][0]	1	
marker_bit	1	1
PMV[1][1][0]	14	
marker_bits	2	3
PMV[1][1][1]	14	
motion_vertical_field_select[1][1]	1	
marker_bit	1	1
coded_block_pattern	12	
marker_bits	4	F
num_mv_bits	8	
num_coef_bits	14	
marker_bits	2	3

【図６９】



トランスコーダ 131

【図４６】

history stream(40-7)

	bits	value
num_other_bits	7	
marker_bit	1	1
}		

【図４７】

history_stream() {	No. of bits	Mnemonic
next_start_code()		
sequence_header()		
sequence_extension()		
extension_and_user_data(0)		
if (nextbits() == group_start_code) {		
group_of_pictures_header()		
extension_and_user_data(1)		
}		
picture_header()		
picture_coding_extension()		
extensions_and_user_data(2)		
picture_data()		
sequence_end_code	32	bslbf
}		

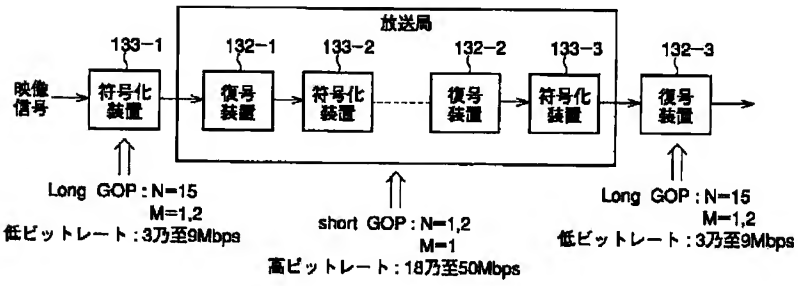
【図４８】

sequence_header() {	No. of bits	Mnemonic
sequence_header_code	32	bslbf
horizontal_size_value	12	uimsbf
vertical_size_value	12	uimsbf
aspect_ratio_information	4	uimsbf
frame_rate_code	4	uimsbf
bit_rate_value	18	uimsbf
marker_bit	1	bslbf
vbv_buffer_size_value	10	uimsbf
constrained_parameters_flag	1	bslbf
load_intra_quantiser_matrix	1	uimsbf
if (load_intra_quantiser_matrix)		
intra_quantiser_matrix[64]	8*64	uimsbf
load_non_intra_quantiser_matrix	1	uimsbf
if (load_non_intra_quantiser_matrix)		
non_intra_quantiser_matrix[64]	8*64	uimsbf
next_start_code()		
}		

【図４９】

sequence_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
profile_and_level_indication	8	uimsbf
progressive_sequence	1	uimsbf
chroma_format	2	uimsbf
horizontal_size_extension	2	uimsbf
vertical_size_extension	2	uimsbf
bit_rate_extension	12	uimsbf
marker_bit	1	bslbf
vbv_buffer_size_extension	8	uimsbf
low_delay	1	uimsbf
frame_rate_extension_n	2	uimsbf
frame_rate_extension_d	5	uimsbf
next_start_code()		
}		

【図７０】



【図53】

picture_header() {	No. of bits	Mnemonic
picture_start_code	32	bslbf
temporal_reference	10	uimsbf
picture_coding_type	3	uimsbf
vbv_delay	16	uimsbf
if (picture_coding_type == 2 || picture_coding_type == 3) {		
full_pel_forward_vector	1	bslbf
forward_f_code	3	bslbf
}		
if (picture_coding_type == 3) {		
full_pel_backward_vector	1	bslbf
backward_f_code	3	bslbf
}		
while (nextbits() == '1') {		
extra_bit_picture /* with the value '1' */	1	uimsbf
extra_information_picture	8	uimsbf
}		
extra_bit_picture /* with the value '0' */	1	uimsbf
next_start_code()		
}		

【図55】

extension_data() {	No. of bits	Mnemonic
while (nextbits()== extension_start_code) {		
extension_start_code	32	bslbf
if (nextbits() == "Quant Matrix Extension ID")		
quant_matrix_extension()		
else if (nextbits() == "Copyright Extension ID")		
copyright_extension()		
else		
picture_display_extension()		
}		
}		

【図５４】

picture_coding_extension() {	No . of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
f_code[0][0] /* forward horizontal */	4	uimsbf
f_code[0][1] /* forward vertical */	4	uimsbf
f_code[1][0] /* backward horizontal */	4	uimsbf
f_code[1][1] /* backward vertical */	4	uimsbf
intra_dc_precision	2	uimsbf
picture_structure	2	uimsbf
top_field_first	1	uimsbf
frame_pred_frame_dct	1	uimsbf
concealment_motion_vectors	1	uimsbf
q_scale_type	1	uimsbf
intra_vlc_format	1	uimsbf
alternate_scan	1	uimsbf
repeat_first_field	1	uimsbf
chroma_420_type	1	uimsbf
progressive_frame	1	uimsbf
composite_display_flag	1	uimsbf
if (composite_display_flag) {		
v_axis	1	uimsbf
field_sequence	3	uimsbf
sub_carrier	1	uimsbf
burst_amplitude	7	uimsbf
sub_carrier_phase	8	uimsbf
}		
next_start_code()		
}		

【図５６】

quant_matrix_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
load_intra_quantiser_matrix	1	uimsbf
if (load_intra_quantiser_matrix)		
intra_quantiser_matrix[64]	8 * 64	uimsbf
load_non_intra_quantiser_matrix	1	uimsbf
if (load_non_intra_quantiser_matrix)		
non_intra_quantiser_matrix[64]	8 * 64	uimsbf
load_chroma_intra_quantiser_matrix	1	uimsbf
if (load_chroma_intra_quantiser_matrix)		
chroma_intra_quantiser_matrix[64]	8 * 64	uimsbf
load_chroma_non_intra_quantiser_matrix	1	uimsbf
if (load_chroma_non_intra_quantiser_matrix)		
chroma_non_intra_quantiser_matrix[64]	8 * 64	uimsbf
next_start_code()		
}		

【図５７】

copyright_extension() {	No. of bits	Mnemonic
extension_start_ ɔde_identifier	4	uimsbf
copyright_flag	1	bslbf
copyright_identifier	8	uimsbf
original_or_copy	1	bslbf
reserved	7	uimsbf
marker_bit	1	bslbf
copyright_number_1	20	uimsbf
marker_bit	1	bslbf
copyright_number_2	22	uimsbf
marker_bit	1	bslbf
copyright_number_3	22	uimsbf
next_start_code()		
}		

【図５８】

picture_display_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
for (i=0; i<number_of_frame_centre_offsets; i++) {		
frame_centre_horizontal_offset	16	simsbf
marker_bit	1	bslbf
frame_centre_vertical_offset	16	simsbf
marker_bit	1	bslbf
}		
next_start_code()		
}		

【図６３】

motion_vectors (s) {	No. of bits	Mnemonic
if (motion_vector_count == 1) {		
if ((mv_format == field) && (dmv != 1))		
motion_vertical_field_select[0][s]	1	uimsbf
motion_vector(0, s)		
} else {		
motion_vertical_field_select[0][s]	1	uimsbf
motion_vector(0, s)		
motion_vertical_field_select[1][s]	1	uimsbf
motion_vector(1, s)		
}		
}		

【図６０】

slice() {	No. of bits	Mnemonic
slice_start_code	32	bslbf
slice_quantiser_scale_code	5	uimsbf
if (nextbits() == '1') {		
intra_slice_flag	1	bslbf
intra_slice	1	uimsbf
reserved_bits	7	uimsbf
while (nextbits() == '1') {		
extra_bit_slice /* with the value '1' */	1	uimsbf
extra_information_slice	8	uimsbf
}		
}		
extra_bit_slice /* with the value '0' */	1	uimsbf
do {		
macroblock()		
} while (nextbits() != '000 0000 0000 0000 0000 0000')		
next_start_code()		
}		

【図６１】

macroblock() {	No. of bits	Mnemonic
while (nextbits() == '0000 0001 000')		
macroblock_escape	11	bslbf
macroblock_address_increment	1-11	vlclbf
macroblock_modes()		
if (macroblock_quant)		
macroblock_quantiser_scale_code	5	uimsbf
if (macroblock_motion_forward ||		
(macroblock_intra && concealment_motion_vectors))		
motion_vectors(0)		
if (macroblock_motion_backward)		
motion_vectors(1)		
if (macroblock_intra && concealment_motion_vectors)		
marker_bit	1	bslbf
}		
}		

【図６２】

macroblock_modes() {	No. of bits	Mnemonic
macroblock_type	1-9	vlclbf
if (macroblock_motion_forward ||		
macroblock_motion_backward) {		
if (picture_structure == 'frame') {		
if (frame_pred_frame_dct == 0)		
frame_motion_type	2	uimsbf
} else {		
field_motion_type	2	uimsbf
}		
}		
if ((picture_structure == "Frame picture") &&		
(frame_pred_frame_dct == 0) &&		
(dct_type_flag == 1)){		
dct_type	1	uimsbf
}		
}		

【図６４】

motion_vector (r, s) {	No. of bits	Mnemonic
motion_code[r][s][0]	1-11	vlclbf
if ((f_code[s][0] != 1) && (motion_code[r][s][0] != 0))		
motion_residual[r][s][0]	1-8	uimsbf
if (dmv == 1)		
dmvector[0]	1-2	vlclbf
motion_code[r][s][1]	1-11	vlclbf
if ((f_code[s][1] != 1) && (motion_code[r][s][1] != 0))		
motion_residual[r][s][1]	1-8	uimsbf
if (dmv == 1)		
dmvector[1]	1-2	vlclbf
}		

【図６５】

macroblock_type VLC code	macroblock_quant	dct_type_flag	macroblock_motion_forward	macroblock_motion_backward	Description
1	0	1	0	0	Intra
01	1	1	0	0	Intra, Quant

【図６６】

macroblock_type VLC code	macroblock_quant	dct_type_flag	macroblock_motion_forward	macroblock_motion_backward	Description
1	0	1	1	0	MC, Coded
01	0	1	0	0	No MC, Coded
001	0	0	0	0	MC, Not Coded
0001 1	0	1	0	0	Intra
0001 0	1	1	1	0	MC, Coded, Quant
0000 1	1	1	0	0	No MC, Coded, Quant
0000 01	1	1	0	0	Intra, Quant

【図６７】

macroblock_type VLC code	macroblock_quant	dct_type_flag	macroblock_motion_forward	macroblock_motion_backward	Description
10	0	0	0	0	Interp, Not Coded
11	0	1	1	1	Interp, Coded
010	0	0	0	0	Bwd, Not Coded
011	0	1	0	1	Bwd, Coded
0010	0	0	0	0	Fwd, Not Coded
0011	0	1	1	0	Fwd, Coded
0001 1	0	1	0	0	Intra
0001 0	1	1	1	1	Interp, Coded, Quant
0000 11	1	1	1	0	Fwd, Coded, Quant
0000 10	1	1	0	1	Bwd, Coded, Quant
0000 01	1	1	0	0	Intra, Quant

フロントページの続き

(72)発明者　三原　寛司
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(72)発明者　村上　芳弘
東京都品川区北品川６丁目７番35号　ソニー株式会社内

Ｆターム(参考)　5C059 MA00 MA05 MA23 MC38 PP05 PP06 PP07 PP16 RB08 RC11 RC31 SS01 SS07 SS11 SS20 SS22 TA25 TA45 TB03 TB04 TC15 TC20 TC41 UA02 UA05 UA39